



jamk.fi

Monitoring and management application for Pre5ence Oy

Nikita Baranov

Bachelor's thesis

March 2018

Technology, communication and transport

Degree Programme in Information Technology

Jyväskylän ammattikorkeakoulu

JAMK University of Applied Sciences

Author(s) Baranov, Nikita	Type of publication Bachelor's thesis	Date March 2018 Language of publication: English
	Number of pages 41	Permission for web publication: X
Title of publication Monitoring and management application for Pre5ence Oy		
Degree programme Information Technology		
Supervisor(s) Luostarinen, Hannu		
Assigned by Pre5ence Oy		
Abstract <p>The objective of the Bachelor's thesis was to develop a web application prototype of a monitoring and management system to be used with a new alarm device currently being developed by Pre5ence Oy. The application was developed using only freely available technologies and libraries.</p> <p>The main requirement for the application was the ability to show the position of the device based on provided coordinates using a web map system (Leaflet.js). In addition, a user and device management as well as login and user authentication system were required to manage device ownership and access rights.</p> <p>The application was developed independently with minimal involvement from Pre5ence using mainly PHP and PostgreSQL. A small amount of JavaScript was also developed to integrate with Leaflet.js and create a simple AJAX functionality.</p> <p>The result of the thesis is a working prototype of an application that showcases data collected by devices using a simple and light UI with the included management system to control device ownership, among other things. All initially discussed functions were completed with addition of new requirements introduced during the final stage of development.</p> <p>The client was pleased with the delivered application. Future development of the application was also discussed. In addition to the application, full documentation and user guides were created to assist with future development.</p>		
Keywords (subjects) web application, web map service, PHP, PostgreSQL, SQL		
Miscellaneous		

Tekijä(t) Baranov, Nikita	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä maaliskuu 2018
		Julkaisun kieli: Englanti
	Sivumäärä 41	Verkojulkaisulupa myönnetty: X
Työn nimi Seuranta- ja hallintajärjestelmä Pre5ence Oy:lle		
Tutkinto-ohjelma Tietotekniikan koulutusohjelma		
Työn ohjaaja(t) Hannu Luostarinen		
Toimeksiantaja(t) Pre5ence Oy		
Tiivistelmä <p>Opinnäytetyössä tehtävänä oli kehittää seuranta- ja hallinnointiverkkosovellus, joka tulisi käyttöön uuden, Pre5ence Oy:n kehityksessä olleen hälytyslaitteen rinnalle. Sovellus piti kehittää käyttäen vapaasti saatavilla olevia tekniikoita ja kirjastoja.</p> <p>Sovelluksen päävaatimuksena oli laitteen sijainnin ilmaiseminen käyttäen verkkokarttapalvelua (Leaflet.js). Tämän lisäksi käyttäjä- ja laitehallinta sekä kirjautuminen ja autentikointijärjestelmät olivat tarpeen, jotta laitteiden omistajuutta sekä käyttäjien oikeuksia voisi hallita.</p> <p>Sovellus oli kehitetty itsenäisesti hyvin vähäisellä toimeksiantajan osallistumisella käyttäen enimmäkseen PHP- sekä PostgreSQL-kieliä. Pieni määrä JavaScript-koodia oli kehitetty sovelluksen integroimiseen Leaflet.js-kirjastoon sekä AJAX-toiminnallisuutta varten.</p> <p>Työn lopputuloksena on toimiva sovellusprototyyppi, joka näyttää laitteiden keräämää tietoa yksinkertaisessa ja kevyessä käyttöliittymässä joka mm. mahdollistaa laitteiden sekä käyttäjien oikeuksien määrittelyä. Kaikki alun perin sovitut toiminnot, sekä työn loppuvaiheessa pyydetty toiminnallisuudet saatiin toteutettua.</p> <p>Toimeksiantaja oli tyytyväinen toimitettuun sovellukseen. Jatkokehityksestä keskusteltiin myös. Sovelluksen lisäksi järjestelmän dokumentaatio sekä käyttöohjeet luotiin jatkokehityksen avuksi.</p>		
Avainsanat (asiasanat) web sovellus, verkko sovellus, kartta palvelu, netti kartta palvelu, PHP, PostgreSQL, SQL		
Muut tiedot		

Contents

Terminology.....	4
1 Introduction	5
1.1 Presence Oy	5
1.2 Motivation and Objective	5
1.3 The author.....	6
2 Tools and technologies.....	6
2.1 Project constraints.....	6
2.2 PHP.....	6
2.2.1 What is PHP	6
2.2.2 PHP Frameworks.....	8
2.2.3 Other server-side programming languages.....	9
2.2.4 PHP vs ASP.NET vs JSP	9
2.3 JavaScript.....	10
2.4 AJAX	11
2.5 HTML and CSS.....	12
2.6 Web Map Service	12
2.6.1 What is a web map service	12
2.6.2 Leaflet.js vs Google Maps	12
2.7 SQL.....	13
2.7.1 What is SQL	13
2.7.2 SQL comparison.....	13
2.7.3 PostgreSQL vs MySQL	14
2.8 Bootstrap	15
2.9 Agile Software Development	15
2.10 Adobe Dreamweaver	16

3	Concerns	16
3.1	Leaflet.js integration.....	16
3.2	Database structure	17
3.3	Date and time-based SQL-queries.....	17
3.4	SQL Injection.....	17
3.5	Session hijacking	18
3.6	Changes in EU legislation	19
4	Project Hound	19
4.1	General functionality.....	19
4.2	User definition	20
5	Development phase	20
5.1	Planning and scheduling	20
5.2	Research of available tools and technologies	22
5.3	Technologies and architecture	22
5.4	Database design	22
5.5	Use case definition.....	24
5.6	Password security and generation	25
5.7	General functionality.....	26
5.8	Development iterations	27
5.8.1	Stage one.....	27
5.8.2	Stage two.....	27
5.8.3	Stage three	27
5.8.4	Stage four	29
5.9	Final functionality	30
5.10	Testing	31
6	Evaluation	31

7	Future development and improvement.....	32
8	Discussion	33
	References	35

Figures

Figure 1. Most common server-side programming languages (adapted from W3Techs.com 2018).	7
Figure 2. Popularity of PHP Frameworks (adapted from Google Trends 2018)	8
Figure 3. Usage of JavaScript for websites (adapted from W3Techs.com 2018).....	11
Figure 4. Database ranking (adapted from DB-engines.com 2018).	14
Figure 5. Example of visible route on a map.	21
Figure 6. System architecture (3-tier)	22
Figure 7. Visual representation of the database.	23
Figure 8. Use-case for system prototype.	24
Figure 9. Sending email with PHP mail.	25
Figure 10. Password regular expression	26
Figure 11. PHP password_verify	26
Figure 12. Basic system logic.....	26
Figure 13. Database class methods.....	28
Figure 14. PostgreSQL to_timestamp function example	29
Figure 15. PHP date_format function example.....	29
Figure 16. JSON generated with PHP	29
Figure 17. AJAX code to fetch new location data.....	30

Terminology

Pregence

Jyväskylä based technology company that also assigned this project.

Haski

Mobile alarm device and also the main product of Pregence Oy.

Library

A collection of pre-developed code.

PHP

One of the most used server-side scripting languages.

JavaScript (JS)

Dynamic front-end scripting language used by over 90% of all websites.

SQL

Structured Query Language. A language used for programming relational database management systems (RDMS).

Front-end

In web-development, front-end refers to the client-side (representation layer) code of the application, it is the code that is executed on the machine of the end user, e.g. HTML and JavaScript.

Back-end

In web-development, back-end refers to the server-side code of a web application, it is the code that is executed strictly on the server that hosts the application, e.g. PHP.

1 Introduction

1.1 Pre5ence Oy

Pre5ence OY is a Jyväskylä based company that in addition to their own products, provides planning services in the area of embedded systems as well as other technological expertise such as MCU-programming, 3D design and circuit board design, to mention a few. The company was established in 2013 and is entirely owned by its employees. (Pre5ence Oy 2018).

The main product of Pre5ence Oy is a mobile 'Haski' alarm device family, which was developed entirely by Pre5ence and is aimed specifically towards Nordic environmental conditions and needs. The device is robust, simple to operate and designed for off-the-grid mobile use. The current Haski family features two device types: Haski and Haski+. These devices are capable of collecting various data of their surroundings and sending SMS messages upon a possibly triggered alarm. Long battery life makes it a practical solution for monitoring 'out-of-the-way' locations such as cottages, garages and boats. (Hälyttimet [Alarm devices] 2018).

1.2 Motivation and Objective

It is safe to say that the Internet of Things (IoT) is here to stay and although there are some concerns as pointed out in an article written by a Symantec employee (5 predictions on the future of the Internet of Things 2018), there are also many reasons to look forward to future developments.

With some 80% of the population of the 'developed world' have an Internet connection. (Global Internet usage 2018) It would be easy to claim that the majority of the people would assume that any type of device has the ability to send and possibly receive data. This data would then be presented to the user for some type of analysis or viewing (e.g. map apps on smartphones).

For mobile alarm devices this is crucial as they are designed to be used on boats, mobile homes as well as other property that may not be connected to any static electricity or connectivity grid and might not be easily accessible. This need to show collected data extends to various utility and storage vehicles.

This thesis project aimed at developing a web-based UI/management application prototype that would be used alongside the new addition to the Haski family. The application was meant to be developed with a minimal budget using only licensing free services, tools and libraries. This project did not aim towards development of device-to-server communication system since that crucial element had already been developed before the project began.

1.3 The author

The author of this thesis has over seven years of experience working for an online marketing company as a web developer; however, he has limited skills in any kind of server-side PHP or client-side JavaScript programming. The author's motivation behind this project was to improve his programming skills as well as to create an application for the upcoming product of Pre5ence.

2 Tools and technologies

2.1 Project constraints

The application was to be developed on a minimal budget, which set some restrictions to possible tools and technologies, however, it did not effectively limit the development in any measurable way.

The purpose of the developed application was commercial, which together with budgetary limitations set restrictions for use of some of the more widely used technologies such as Google Maps.

The application was developed on MacBook Pro and tested locally before uploading to the server. The server platform was organized from Digital Ocean by Pre5ence and featured an Ubuntu 16.x operating system. The server setup required only minimal configuration and was mainly handled by Pre5ence staff.

2.2 PHP

2.2.1 What is PHP

PHP is over a 20-year old open-source server-side scripting language that is often used for web development and is still being actively used and developed. (PHP

Manual: Preface 2018). Based on recent statistics by W3Tech.com (see Figure 1), it is used in a vast majority of all websites today.

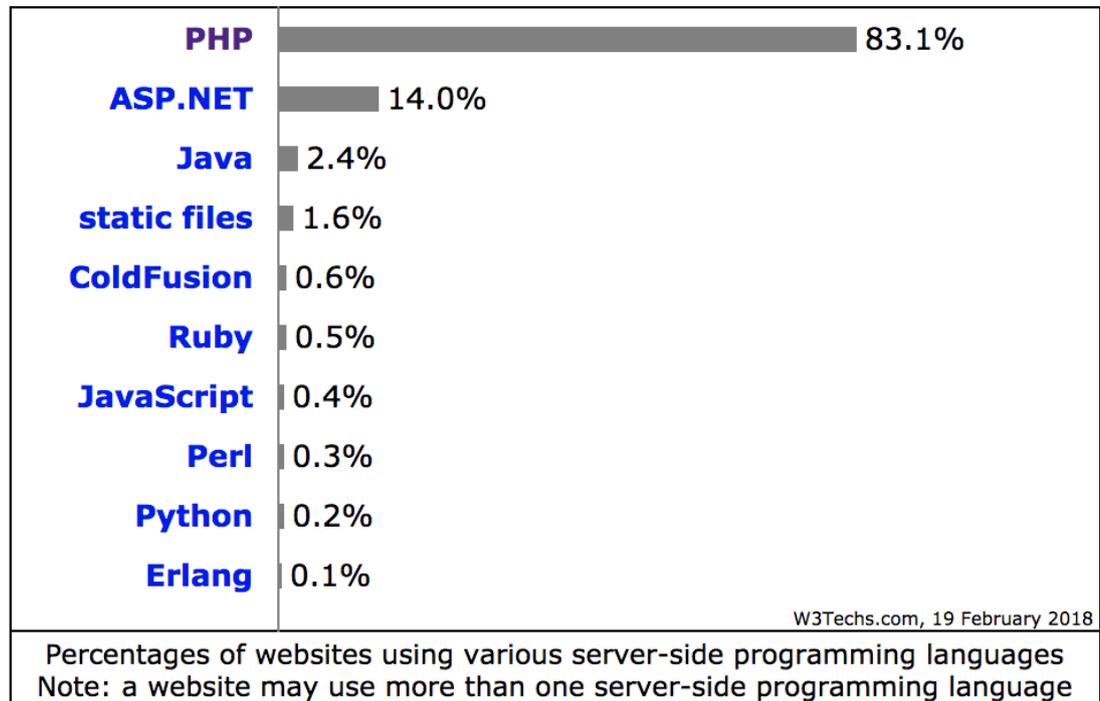


Figure 1. Most common server-side programming languages (adapted from W3Techs.com 2018).

One of the most important things to know when programming with PHP is that by default, it is a stateless language. In other words, a PHP application does not 'remember' the user or user's actions. This is problematic when an application requires a dynamic approach, as in the case of this project.

There are, however, few ways to bypass this feature. Information such as values of variables can be passed via GET, POST and SESSION as well as other 'superglobal' variables. (PHP Superglobals 2018).

GET and POST are most commonly used with some type of forms and are transferred via HTTP (and HTTPS) protocol, while SESSION is often used as a temporary storage for data that is used to identify a user and is stored on the server that hosts the application.

Using either of these methods to pass variables later used in the script has a potential security issue, so any data passed through them needs to be either

sanitized or used in a manner that does not compromise the security of the application.

In this project, the choice of using PHP was almost self-evident: it is free, light, easy to setup and extremely popular. Consequently, it also has a tremendous amount of free tutorials and guides. Having almost unlimited access to examples, code-snippets, discussions and tutorials made PHP stand out as practically the only feasible solution for this project due to the author's limited programming know-how as well as budgetary limitations.

2.2.2 PHP Frameworks

Currently there are over 20 different PHP frameworks with Laravel being clearly the most popular as illustrated in Figure 2. (Most searched for PHP frameworks 2018). While this data does not show the actual usage of any of these frameworks, it does provide a very clear message regarding their popularity and especially interest in them for the past few years.

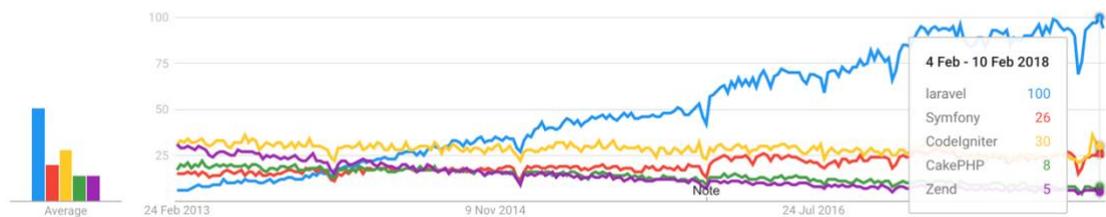


Figure 2. Popularity of PHP Frameworks (adapted from Google Trends 2018)

Whether to use a framework or not often depends on the scope of the project. Because of the quite highly developed ready-to-use code, using a framework can significantly decrease development time and offer a fast and secure development environment. On the other hand, a framework can also create unnecessary complexity when the development requires more specific code and when code-sharing is not necessary. (Should You Use a PHP Framework? Five Pros and Cons 2018).

The primary reason to avoid frameworks in this project was that the efficient use of a framework would require investing substantial time in learning how to properly

use it. There was also no need to share the code as it was developed by a single developer. In addition, using a framework would be a very permanent solution as written code would likely be only compatible with the framework of choice.

As a consequence, the developed application is substantially lighter than it would be if a framework was used and does not require familiarity with any PHP framework.

2.2.3 Other server-side programming languages

PHP is not the only scripting / programming language available when developing server-side code for an application, in fact, there are over 20 others and many of them are free /open-source.

Currently the most common ones are PHP, ASP.NET and Java, PHP being clearly the most commonly used. (Usage of server-side programming languages for websites 2018).

2.2.4 PHP vs ASP.NET vs JSP

Comparing PHP to ASP.NET or JSP is not as simple as it might seem. Unlike PHP, ASP.NET as well as JSP are not actual programming languages. ASP.NET is rather a web-framework (much like a PHP framework) and JSP is rather a technology based on Java servlets with a strong connection to Java libraries.

Language wise, PHP is mostly based on C and Perl while ASP.NET uses C# and JSP is based on Java. Unlike PHP, ASP.NET and JSP are also fully object-oriented languages; PHP however, gained full OOP capability only starting with PHP5 and is still not considered to be as object oriented as the other two systems.

Cost wise, none of the three have significant difference in regard to available tools or even servers. ASP.NET; however, being a product of Microsoft requires a Microsoft Windows Server, which is often a slightly more expensive hosting solution than a Linux based one. JSP, just like PHP, is not limited to any specific platform but does require a Java servlet friendly system such as Apache Tomcat.

Performance wise, PHP is clearly the most efficient due to its lightness and due to it being primarily developed for one single purpose; web-development. Both PHP and

JSP are web-technologies, however, as the complexity of the application grows, PHP can become cumbersome while JSP and ASP.NET can rely on a vast number of included libraries. Both ASP.NET and JSP are often used for enterprise solutions, largely due to their ready-made libraries and thus more unified code.

Perhaps the greatest single difference between the three, apart from the syntax, is the community and availability of various guides, examples, libraries and ready-made open-source applications. From that point of view PHP offers vastly greater opportunities especially for beginners and is often regarded as a more easily approachable solution.

Another significant difference is that ASP.NET can be used to develop non-web applications such as desktop and mobile applications while PHP has very limited usage outside web-application development. JSP, while being primarily web-programming technology relies heavily on Java, which is often used for e.g. mobile application development and is also a part of J2EE-platform.

Developing this project using ASP.NET or JSP would have required a different server solution from the one already implemented by Pre5ence. In addition, it would have required re-developing some of the already created scripts to a new platform as they were developed with PHP.

Because of the extensive amount of examples and tutorials as well as its generally more beginner friendly approach, PHP was considered as the most viable solution for this project. While PHP was clearly the ideal solution, there is no reason to doubt that the same application functionality could not have been achieved by ASP.NET or JSP.

2.3 JavaScript

JavaScript is a scripting language that is often used for creation of dynamic functionality of a website. It is currently the most used front-end scripting language and is used by over 90% of all websites (see Figure 3). (Usage of JavaScript for websites 2018). While JavaScript is most often used on the client-side of the application, it can also be used as a server-side scripting language. JavaScript was primarily used as a part of Leaflet.js and AJAX script used by the application.

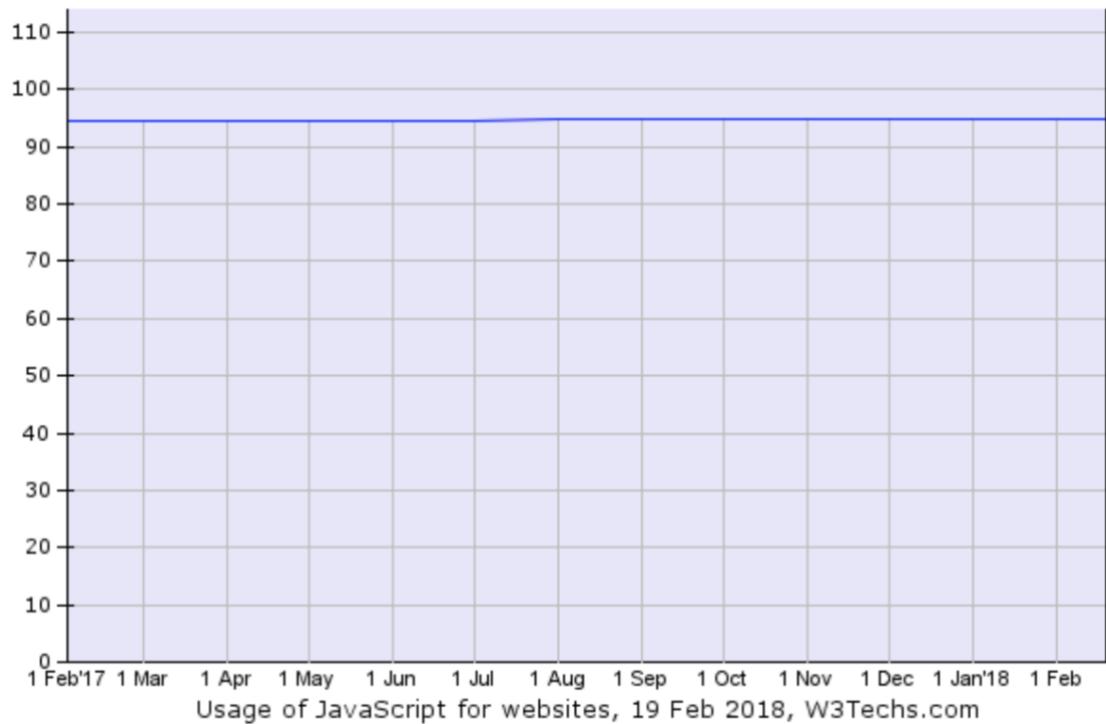


Figure 3. Usage of JavaScript for websites (adapted from W3Techs.com 2018).

2.4 AJAX

AJAX refers to Asynchronous JavaScript And XMLHttpRequest. It is a web-development technique that allows for an asynchronous data retrieval from the server without interfering with display or behavior of the existing page. (Ajax (programming) 2018). This allows for a creation of very dynamic web pages and web applications that can communicate with the server and database without a need to 'reload' a page.

While AJAX provides great dynamic functionality, one of its drawbacks is that it relies on support and availability of JavaScript or XMLHttpRequest by the browser that is used when accessing the web application. In addition, using AJAX can lead to complicated code that is difficult to maintain and even test. (Ajax (programming) 2018).

AJAX was required for only one part of this project, Live Map-view that automatically updates the position of the device on the map and draws a route between retrieved locations in 'real-time'.

2.5 HTML and CSS

HTML (Hypertext Markup Language) is a standard markup language used for creating web pages and web applications. (HTML 2018) Its appearance is further enhanced with CSS and JavaScript. HTML code is what the browser-software used to access the website translates into a visible non-code representation of a web page or application.

CSS (Cascading Style Sheets) is a language describing the presentation of a document written in a markup language. While CSS is most often used with HTML, it can also be used with other markup languages, e.g. XML. (What is CSS 2018).

2.6 Web Map Service

2.6.1 What is a web map service

Web Map Service (WMS) refers to a protocol that is used to provide georeferenced map images, which are generated using a geographic information system (GIS). (Web Map Service 2018.)

WMS protocol is currently standardized by Open Geospatial Consortium, which is an International industry consortium featuring companies, governmental agencies as well as universities. (About OGC 2018.)

WMS is used by a plethora of various map systems that provide web map functionality, e.g. TomTom, Google and Garmin.

2.6.2 Leaflet.js vs Google Maps

Leaflet.js is a light, open-source JavaScript library for mobile-friendly interactive maps. (Leaflet.js overview 2018). The simplicity and specifically the quality of documentation makes it extremely easy to understand even with relatively low JavaScript programming skills.

Google Maps is an advanced web map service provided by Google that is able to provide 'top-down' as well as 'street'-view of selected area. Google provides

extensive documentation and cross-platform usability via its GoogleMaps API system. (Google Maps 2018.)

Both systems are relatively easy to comprehend and use, although full functionality of Google Maps can only be achieved through its various API systems, which are generally heavier than scripts provided by Leaflet.js.

The main difference between the two is that Leaflet.js is a JavaScript library while Google Maps is a full web map service that offers code as well as imagery.

Cost wise, Google Maps offers only a limited free usability that is limited to 25,000 map loads per day. Leaflet.js itself is completely open source and does not require any licensing, however, it requires an external map imagery provider such as OpenStreetMaps or similar.

One of the primary deciding factors was the cost of the initial setup. While Google Maps would have been a much more powerful system, its usage outside development would have created substantial costs almost immediately. Using Leaflet.js, however, would allow relatively easy change in map imagery provider and thus control possible map imagery related costs.

Because of the nature of this project, OpenStreetMaps was chosen as map imagery provider as it had the friendliest use policy. (Copyright and License 2018.)

2.7 SQL

2.7.1 What is SQL

SQL refers to Structured Query Language, which is a programming language used for managing data stored in relational database management systems (RDBMS). (SQL 2018). "SQL is designed for a specific purpose: to query data contained in a relational database. (ibid.)"

2.7.2 SQL comparison

According to data collected by db-engines.com between 2013 and 2018 (DB-Engines Ranking 2018), there are over 300 database management systems (DBMS), out of which the five most popular are Oracle, MySQL, Microsoft SQL server, PostgreSQL

and MongoDB (see Figure 4).

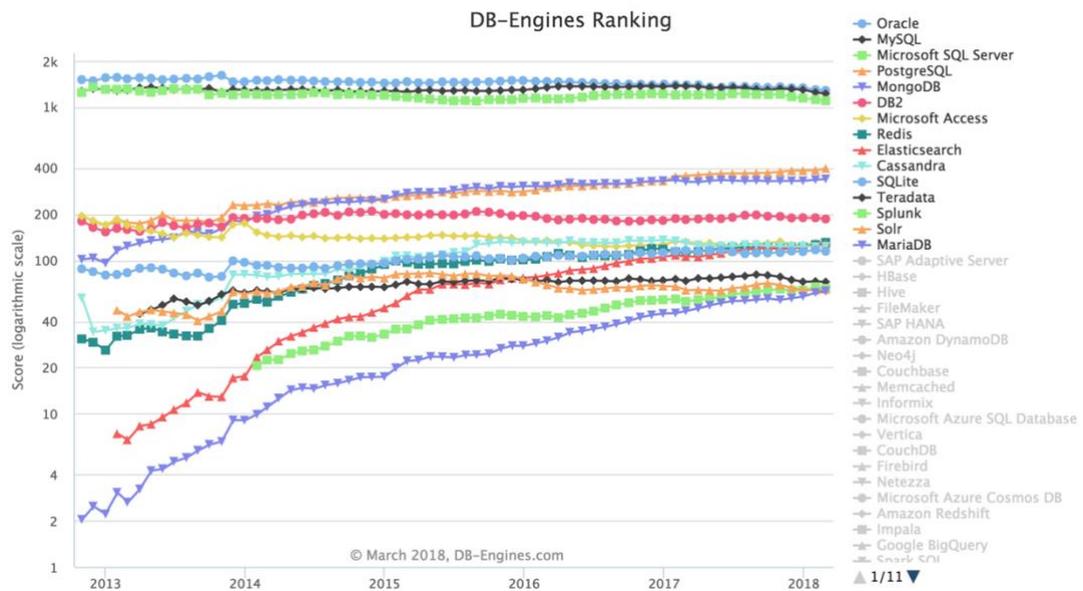


Figure 4. Database ranking (adapted from DB-engines.com 2018).

The top five systems are all feature rich and can easily fulfill the needs of the intended system. Their licensing, however, varies and that played a major role in selecting DBMS for this application. (Microsoft SQL Server vs. MongoDB vs. MySQL vs. Oracle vs. PostgreSQL 2018.)

2.7.3 PostgreSQL vs MySQL

PostgreSQL is an open source object-relational database system which is actively developed and considered to be the most advanced and SQL compliant DBMS today. MySQL is currently the most popular DBMS and although owned by Oracle, it is also an open-source system. Both are not restricted to a specific operating system and are freely available for almost any use. MySQL is also the default database system used in LAMP (Linux, Apache, MySQL, PHP) setup. (DBMS comparison 2018).

PostgreSQL and MySQL both have extensive documentation. PostgreSQL differs from other similar systems, including MySQL, by having no commercial use license. While MySQL is indeed an open-source system and offers a GPL license that allows its use even for commercial purposes, it also features a separate commercial licensing that is required if MySQL is being sold as a part of non-free program or application. (Schwartz, B. 2009)

Capability and performance wise both systems are considered to be on the same level, however, PostgreSQL is more actively developed, more compliant with SQL standard and has complete support for ACID concept (Atomicity, Consistency, Isolation, Durability) that ensures reliable transactions. (Tezer 2014.)

Choosing the database engine required consideration of possible future needs and the direction further development could take. Because the nature of the application required quick write-read speeds as well as possible need for concurrency as well as likelihood of complexity of SQL queries, PostgreSQL was eventually selected as the database system for this project.

2.8 Bootstrap

Bootstrap is a free and open-source front-end library for websites and web applications. Initially developed at Twitter, it has long since become an independent front-end framework. (Bootstrap History 2018).

Bootstrap was found while searching for simple to use and free CSS solution to avoid spending time developing CSS code for the application. This was mainly due to not having any set specifications nor requirements regarding UI/UX of the application.

An article from keycdn.com. (Top 10 Front-End Frameworks of 2016 2018) was used as initial guide to compare available frameworks and determine their suitability for this project.

Bootstrap was not chosen because of a specific need or requirement, however, rather because of its extensive documentation and convenient features.

2.9 Agile Software Development

Agile software development, also often known simply as Agile development, is a development process in which requirements as well as solutions evolve throughout the process. In practice, this means that the planning takes into account that not all issues and needs may be known when the development process starts. (Agile software development 2018).

This approach is not aimed strictly at software development and can be used for many other industries and needs.

There are many variations of agile development, however, the fundamental idea is that the development process is broken up into smaller parts. Each part is often referred to as an iteration and each iteration has a specific goal (e.g. a function) that needs to be reached before the next iteration starts.

While this approach was not specifically chosen at the beginning of the project, it quickly evolved into it, because it supported the learning process during the development. Because of the nature of the project, agile was also a natural choice.

2.10 Adobe Dreamweaver

Adobe Dreamweaver is a part of the Adobe Creative Cloud software solution provided by Adobe Systems Inc. Dreamweaver is aimed at developers as well as designers and offers a wide range of functionality. (Adobe Dreamweaver 2018).

Dreamweaver was chosen for a few simple reasons: the author already had it, had used it for over a decade and found it easy to use as well as non-restricting.

The program of choice in this case was very cosmetic and had no impact on the written code. In fact, there are several other, very similar software solutions that do almost exactly the same thing in an almost identical way.

Some of the other similar software solutions are listed below:

- Microsoft Visual Studio
- Notepad++
- Komodo IDE
- NetBeans

3 Concerns

3.1 Leaflet.js integration

The main concern with Leaflet.js integration was the ability to pull coordinate information for a specific device for a specific time and use found information to

correctly place a 'marker' on a map. This concern was mainly due to lack of programming skills and experience in PHP and JavaScript.

3.2 Database structure

The concern regarding database structure was mainly due to the need for a database structure that would allow finding the correct information to show to the user and to the need to be able to define correct ownership and access rights between users and devices. In addition, such structure had to take into account possible contract related restrictions and possible ownership changes.

3.3 Date and time-based SQL-queries

Perhaps the greatest concern was only realized after the development of the application already started. The ability to search information based on stored date and time information was completely new to the developer and required further research into PostgreSQL and PHP capability to handle and manipulate date and time related data.

3.4 SQL Injection

SQL Injection is one of the most common methods used for hacking and breaking the intended functionality of a database driven website/application. (SQL Injection 2018).

This vulnerability is often based on an insecure way of handling data provided by the user via some type of a web form. (SQL Injection Tutorial 2018). Because it relies on the actual communication between client and server, this vulnerability exists across web-technologies and has to be taken into account when any database driven application is being developed, irrelevant from used database server.

There are multiple ways to protect the application from potential SQL injection attacks. Data sanitization is one and that should be done regardless of whether or not any database system is used.

Perhaps the best approach, however, is to use prepared SQL statements. A prepared SQL statement is one that has a pre-defined structure that cannot be

changed without having a direct access to the source code of the application. In a prepared SQL statement, parameter values can be bound to specific locations of the statement and defined later. Apart from being an efficient SQL injection prevention mechanism, using prepared statements has other advantages as well (PHP prepared statements 2018).

3.5 Session hijacking

In web-applications a session refers to a server-side storage of data relevant to a particular client-server connection. As a session is stored on the server, it requires a way to connect a relevant data to the relevant connection. This is done via session cookies that are used to identify which stored data is relevant for which connection. This session identification information, often called a 'session id', is passed via HTTP protocol and is thus potentially exploitable.

Sessions are often used as authentication methods for websites and applications and because of this, they are a prime target for any potential attacker.

Session hijacking is an exploitation attack in which attacker attempts to access session specific data of another, already authenticated user. (Session hijacking attack 2018).

There are multiple ways to attempt to prevent session hijacking, however, none of them are 100% secure because identifier information still needs to be communicated between client and server. Securing a connection with SSL is one of the most effective ways as it creates a very effective 'pipe' through which communication between client and server happens. Other ways should be used as well, however, for example, regenerating session id at certain steps of application, setting cookie lifetime and renaming default session cookie.

For this project, session hijacking was a real concern as there was a clear need to store information to identify and authenticate user.

3.6 Changes in EU legislation

To achieve the requested functionality, some user related information would need to be saved within the system, e.g. email addresses and names.

To avoid possible future complications, the system had to be developed in such a way that it would require the minimal user related information, in this case it would be an email address and possibly the name of the client (e.g. business or company name). This system would essentially be a closed system that would require action by the system administrator before any client is able to join and add devices or introduce new users.

Any user related information would also need to be easily removed and only the user, the owner of the user (the client) and the administrator would be able access user related information.

Deeper consideration of effects of the change in legislation could not be done at this point, as this subject was not yet researched by Presence.

4 Project Hound

4.1 General functionality

The basic idea for the system was that it would require some type of login page, after which a user could access the device information of allowed devices. The user type structure would need to have several tiers to allow for 'ownership' of devices and management of users. Future needs could also require a functionality based on the type of the contract.

The main functions of the application were set on the ability to show the location of the device using a web map service and based on received coordinate data. Some other relevant information provided by the device was meant to be shown as well.

A user and device management system was also required to manage the user and the device ownerships and to provide a testing platform that would give a better

insight regarding the relevance of the information shown in the device log tables as well as possible changes that would need to be done to the device itself.

4.2 User definition

A potential user of the system could be an owner of a boat or simply an employee of a security company assigned to monitor a certain location/vehicle. Because of the versatility of users, this application was developed to primarily cater for the needs of two types of users:

1. Client user, to manage and view owned devices and to manage own users.
2. Basic user, to view allowed devices and their data.

In addition to this, further availability of certain functions within the system could be adjusted through contracts that could be created and edited by the system administrator.

5 Development phase

5.1 Planning and scheduling

The project had two distinctive parts that had to be developed:

1. Integration with Leaflet.js
2. UI / management system for users and devices

An initial functionality list was limited to placing location markers and drawing a route between them (see Figure 5), however, a rough management system to manipulate users and devices was also required.

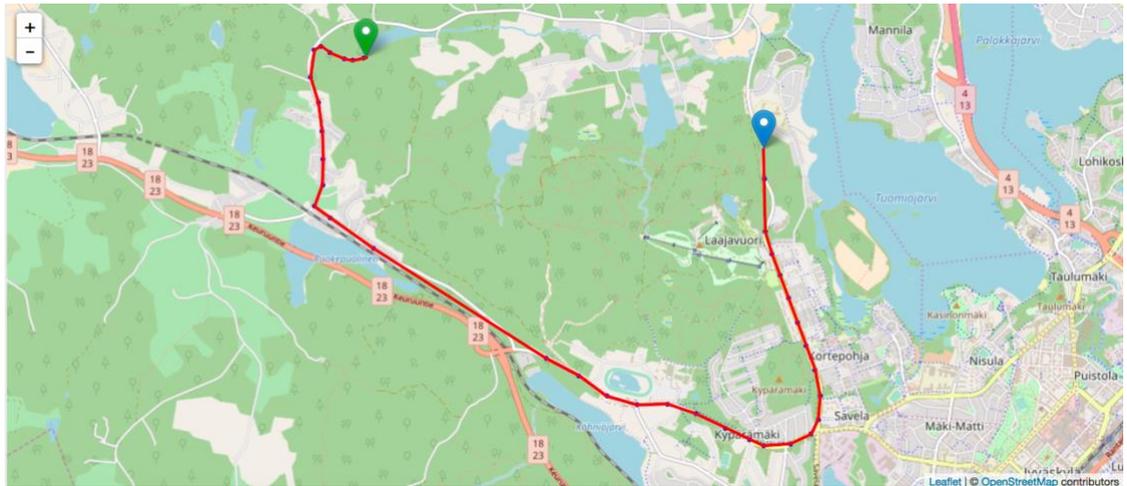


Figure 5. Example of visible route on a map.

The project started with initial planning of the schedule as well as deciding the iterations, their objectives and content. While the overall planning was solely the responsibility of the author, it did require some input from PreSense as the project schedule had to fit in with their business goals and conform to the development schedule of the company. This was achieved with several meetings between the author and the representatives of PreSense throughout the development project.

During the first meeting, a list of required functionalities was created. These requirements had to be completed and would act as the initial proof of concept, proof of the ability of the developer as well as proof that the selected tools would allow further development of the project and would allow for creation of the requested application and functionality.

Further meetings were held to verify and discuss various changes and additions to the functionality of the application based on test results.

Based on provided information and required functionality a prototype use-case was developed (see Figure 8). This was only used for the initial prototype stage of the project. Additional use-cases were created later in the project to cater for needs of other user types and functionality that was not introduced at the planning stage of the project.

5.2 Research of available tools and technologies

The research into selected tools and technologies was done before and during the first stage of the development. This was necessary due to lack of understanding of capabilities of the new device itself and to better understand the needs of the application in general.

The research was mainly focused on three distinctive technologies; PHP, PostgreSQL and Leaflet.js.

5.3 Technologies and architecture

The application was developed to comply with 3-tier architecture (see Figure 6) and to run on an Ubuntu – Linux 16.04 with PHP 7.0.22 and PostgreSQL 9.5.10 and using a HTTPS protocol for secure connection between client and server. All tiers of the applications are handled by the same server.

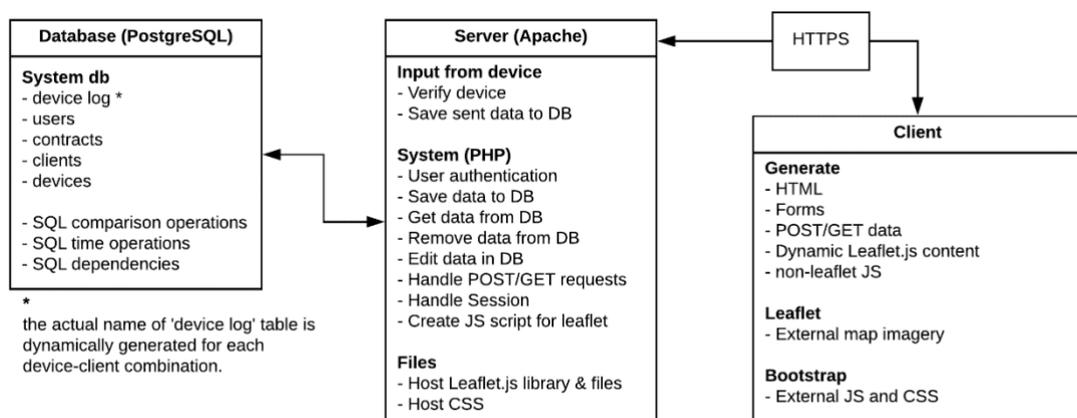


Figure 6. System architecture (3-tier)

5.4 Database design

Designing the database structure of the application can be seen as one of the most fundamental and often critical parts of the database driven application development. This is because solutions and choices made during this time can dictate and even restrict the development of the logical (code) side of the application; however, if done correctly, the designed database can contain automatic functionality within the database system itself (e.g. automatically remove

rows if foreign key connection is broken), which would remove the need of implementing the same functionality with code.

Database design is often done by creating an ER diagram (Entity-Relation diagram) of the database, which shows not only the tables of the database but also datatypes within the tables as well as connections (relations) between tables.

The database for this project was designed with a web-based database design tool from dbDesigner.net, which also allowed automatic SQL code generation of the database and thus considerably simplified the overall design and development process of the database.

Because of the sheer volume of data provided by Haski devices (environmental, location), the database was at the core of the application. There were two fundamentally different needs that had to be taken into consideration: Management of ownership of devices and users and storage of data provided by the devices themselves. While the former required multiple tables and connections, the latter required a simpler approach; however, it had to take into account possible ownership changes (e.g. What if a device is re-sold?).

Because of these needs, it was agreed that device data would be stored in dynamically created tables instead of all-in-one approach that was originally considered. The developed database structure can be seen in Figure 7: the log-table represents the dynamically created table that would be created for each client-device combination.

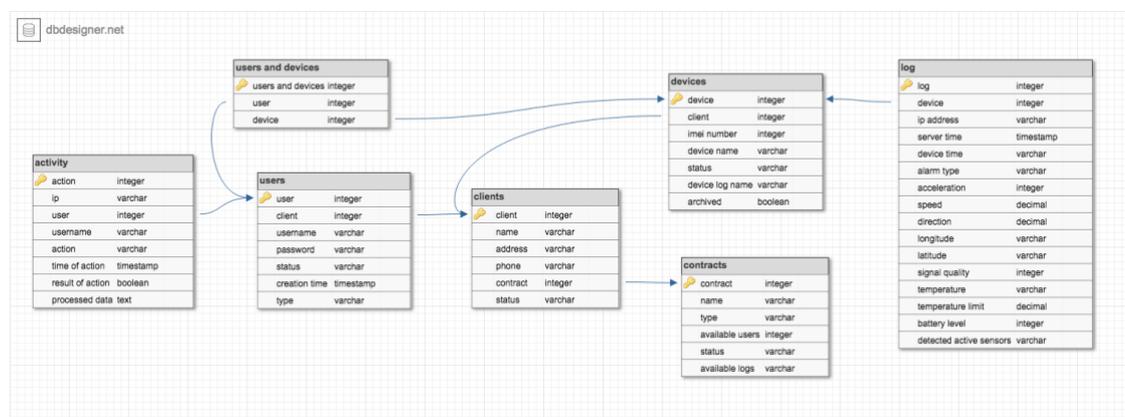


Figure 7. Visual representation of the database.

This structure would also allow for a stricter data control and could be easily further developed or reconfigured to add even more restrictions regarding what data a user can retrieve from the system. In addition, this approach simplified retrieval of the data, allowed for quicker data retrieval and made removal of old data easier.

5.5 Use case definition

Use-case diagrams are used in software and system engineering as a way to represent possible actions users can make within the system. It is an important stage of development because it allows for creation of an abstract functionality of the application. It also fits well with agile development because of its iterative nature. (Use case 2018).

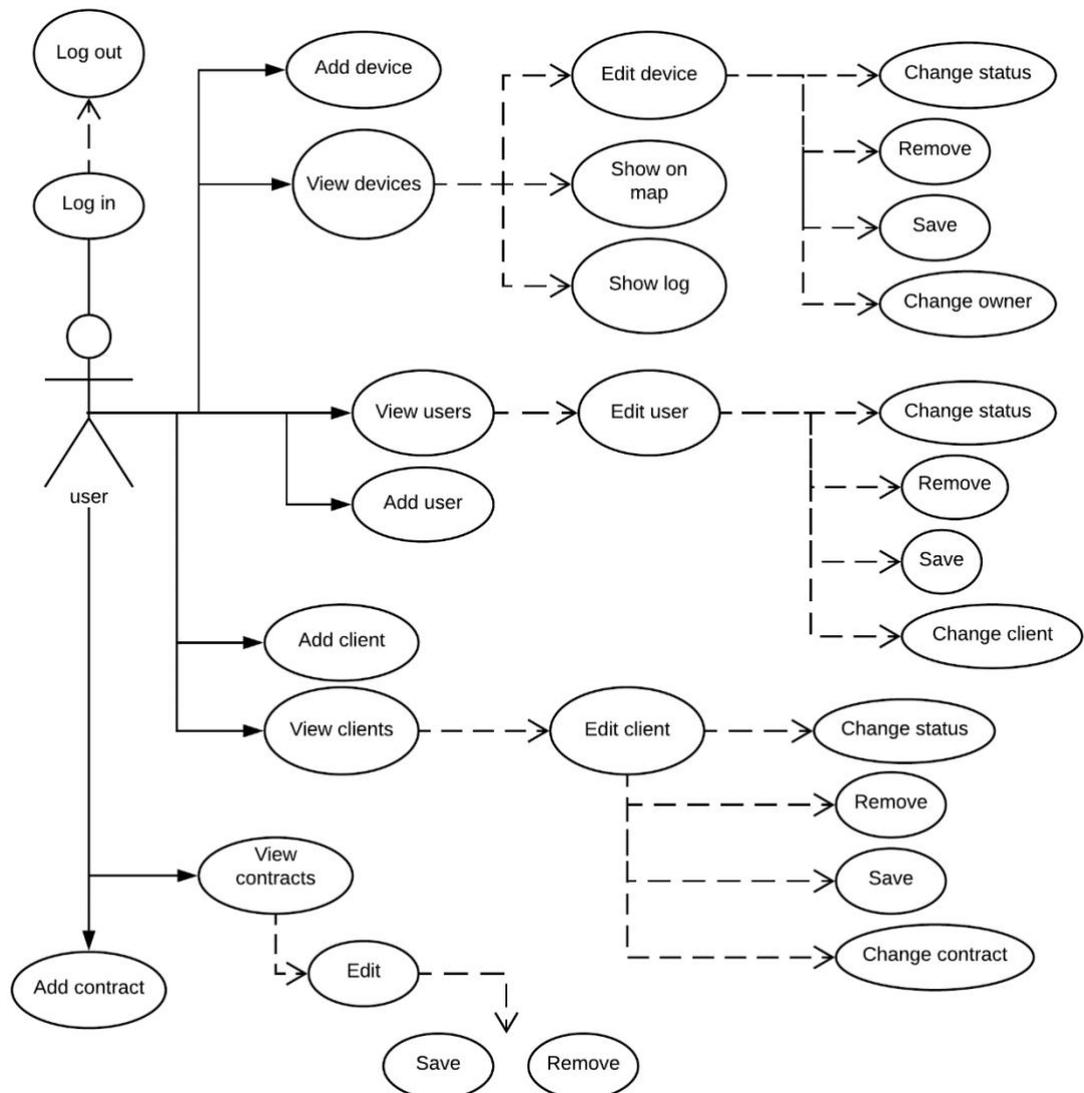


Figure 8. Use-case for system prototype.

5.6 Password security and generation

The application was developed to use a login system for user authentication, which required a secure password creation as well as password reset functionality inside and outside of the system.

While the basic reset functionality within the system itself was straightforward, the same functionality outside of the system required a more sophisticated and a secure way to allow password reset which also had to take into account that the user might not know the previous password. This was achieved by utilizing the native PHP mail functionality which would send password reset link to the provided email if the user was found in the system (see Figure 9). To prevent misuse of the password reset functionality, a system was developed to check for requested user and to generate a 'token' that would be checked against upon attempted access to password reset page. This would mean that any attempt to reset the password outside of the system itself would require access to the email used by attempted user.

```
// subject
$subject = 'Salasanan uusiminen.';
// who is sender
$sender = 'Name of sender'.<email address of sender>;
// who will receive email
$user = $user['username'];
// the message
$msg = "Hei $user,\n\nKlikkaa alla olevaa linkkiä tai kopioi se selaimen osoitekenttään vaihdaaksesi salasanan.";
// url of the link with reset token information
$msg .= "\nLinkki: https://url.for.password.reset?token=".$secret_token;

// verify user and sender are still valid, check that no row change has been passed (just in case)
if ( preg_match( "[\r\n]/", $user ) || preg_match( "[\r\n]/", $sender ) ) {
    // log error
    $log_action->writeLog('Request to reset pwd', ''.$passed_user_name.', username does not match RegEx, email NOT SENT!', 'false');
}
else{
    // attempt to send email to user with subject and message to sender
    if(mail($user,$subject,$msg, "From: ".$sender)){
        // write email sent log
        $log_action->writeLog('Request to reset pwd (step 3/3)', ''.$user.', email sent', 'TRUE');
    }
    else{
        // write email fail log
        $log_action->writeLog('Request to reset pwd (step 3/3)', ''.$user.', email NOT sent', 'false');
    }
}
}
```

Figure 9. Sending email with PHP mail.

Password generation utilizes native PHP hash functionality to secure provided password input. In addition to this, provided password needs to comply with specific length and character requirements (see Figure 10).

```
// matches reg ex for pwd
if($data == 'password'){
    !preg_match("/(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%\-_\/\!#£%^&]).{6,20}/", $data_to_test) ?
        $data_test_result = 'fail'
        : $data_test_result = 'success';
}
```

Figure 10. Password regular expression

Verification of password is done in a similar fashion by comparing possibly found password of attempting user to the one that was passed via login form (see Figure 11).

```
password_verify($attempted_password, $password_retrieved_from_db)
```

Figure 11. PHP password_verify

5.7 General functionality

The functionality of the developed classes can be divided into three sub-types: initiation, data-retrieval and data-returning. The only public function within the classes is responsible for the logic that decides which method needs to be initiated, after this, data-returning methods use data-retrieval methods to fetch data from the database and return the fully HTML formatted result back to the page that initiated and called for the object in question. An abstract representation of basic system logic can be seen in Figure 12.

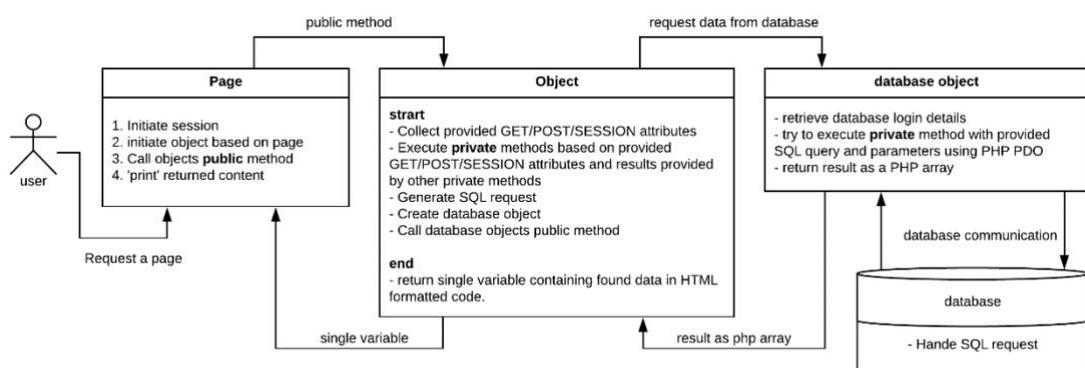


Figure 12. Basic system logic.

The classes were created based on the needed functionality of each part of the system (e.g. users, devices, contracts). This approach allows for a quick reconfiguration if the format of the data needs to be changed later or if data needs

to be passed for some other use (e.g. conversion to JSON data) or even to some other system.

5.8 Development iterations

5.8.1 Stage one

The requirements for the first stage of the development were purposefully left relatively vague and open-ended as the main need was to create a general prototype that would prove that the selected tools and technologies could indeed accommodate the need and would work with the already developed systems.

The overall scope of the application was still relatively unknown at this stage, because of this, this stage was used to get a deeper understanding about the needs of the application and the device itself.

5.8.2 Stage two

The requirements for the second stage were set during the proof of concept meeting at the end of stage one.

At this point, it was relevant to create a system that would also allow for testing, not only of the application itself but of the data provided by the device. The purpose of this stage was to determine if the data provided by the device was enough to accommodate the needs of the planned application.

Based on the results of testing and development of stage two, some changes were made to the device to provide more detailed information regarding types of the alarm and alarm triggers.

As a new feature, the ability to show a route between found coordinates was added to the main feature list.

5.8.3 Stage three

The third stage was used to refine SQL queries and to restrict access rights based on user type and ownership. SQL injection prevention was the main focus of this iteration.

During the third stage, the application was rewritten to conform with OOP methodology as it would significantly simplify the future development of the application. This was also necessary because it simplified and secured the database connection used by various functions.

A special class was created for handling all SQL queries through a PHP data object (PDO) (see Figure 13). This class would also act as the only access point to the database. The returned data would be in a PHP 'associated array' datatype and thus directly available for further analysis or modification by the system.

```

public function databaseConnection() {
    try {
        $this->_postgres_db_connection = new PDO($pdo);

        return $this->databaseQuery($this->_sqlquery, $this->_queryparameters, $this->_typeofquery);
    }
    catch (PDOException $ex){
        $this->_log_error->writeLog('DB Exception', $ex->getMessage().', (query failed)', 'False');
    }
}

private function databaseQuery($sqlquery, $queryparameters, $typeofquery) {
    $found_rows = 'no results found';
    $pdo_statement = $this->_postgres_db_connection->prepare($sqlquery);
    if(count($queryparameters) >= 1){
        $pdo_statement->execute($queryparameters);
    }
    else{
        $pdo_statement->execute();
    }
    if($pdo_statement->rowCount() > 0){
        $found_rows = 'updated';
        if($typeofquery == 'one'){
            $found_rows = $pdo_statement->fetch(PDO::FETCH_ASSOC);
        }
        if($typeofquery == 'all'){
            $found_rows = $pdo_statement->fetchAll(PDO::FETCH_ASSOC);
        }
    }
    $pdo_statement->closeCursor();
    return $found_rows;
}

```

Figure 13. Database class methods.

Some fundamental changes were made to the logic of the application as well, which allowed for searching for 'device log' information based on recorded date and time, and pre-programmed functionality of PHP was utilized to convert the provided data into PostgreSQL recognized timestamp-data. The snippets of the developed SQL and PHP code featuring this can be seen in Figure 14 and Figure 15. Figure 14 shows conversion of non-SQL-timestamp data in the database to a PostgreSQL recognized timestamp and compares it to a PHP generated date-time-combo shown in Figure 15.

```

to_timestamp(device_log_time, 'DD.MM.YYYY HH24:MI:SS') >= to_timestamp(:start_date, 'YYYY-MM-DD HH24:MI:SS')
AND
to_timestamp(device_log_time, 'DD.MM.YYYY HH24:MI:SS') <= to_timestamp(:end_date, 'YYYY-MM-DD HH24:MI:SS')
)

```

Figure 14. PostgreSQL to_timestamp function example

```

':start_date'=>date_format($start_time, "Y-m-d H:i:s"),
':end_date'=>date_format($end_time, "Y-m-d H:i:s"));

```

Figure 15. PHP date_format function example.

5.8.4 Stage four

The fourth and the last stage was mainly used for cleanup and documentation of the application.

As a new feature, a Live-view was requested and delivered. Unlike previous features, this one required the use AJAX technology. The utilization of AJAX was already considered in during the research stage of the project; however, until this point there was very little need for it.

Due to already developed methods to retrieve location data from database using OOP, development of 'Live-map' feature was relatively easy and also confirmed that converting the developed system to OOP was indeed the correct choice, even though it delayed development by two weeks.

Figure 16 shows a snippet of PHP which generates and returns a JSON array. The data used in the array is fetched from a database based on parameters passed via URL and require user to be logged into the system to function. This data is requested by AJAX function and is used to populate variables within Leaflet.js necessary for device location representation on the map.

```

$location = '{"haskitime":"' . $new_location['haski_time'] . '",
            "gps_lat":"' . $new_location['gps_lat'] . '",
            "gps_lon":"' . $new_location['gps_lon'] . '",
            "alarm_type":"' . $this->returnAlarmType($new_location['alarm_type']) . '",
            "temperature":"' . $new_location['temperature'] . '",
            "speed":"' . $new_location['speed'] . '"}';
return $location;

```

Figure 16. JSON generated with PHP

Figure 17 features a snippet of AJAX code that was created to fetch new location information every 16 seconds and initiate a drawRoute() function with a new set of data. The code is generated with PHP and 'printed' to the HTML page upon request; this is only done on page load after which AJAX function is responsible for providing and initiating the drawRoute() function automatically with new location data.

```

$map .= "\n\t\t". 'function ajax(url, fn) {
    try {
        var xhr = new XMLHttpRequest();
    } catch(e) {
        alert(e);
    }
};
$map .= "\n\t\t". 'xhr.open("GET", url, true);';
$map .= "\n\t\t". 'xhr.send();';
$map .= "\n\t\t". 'xhr.onreadystatechange = function() {';
$map .= "\n\t\t". 'if(xhr.readyState == 4 && xhr.status==200) {'
        fn(xhr.responseText);
    }
}
}';

$map .= "\n\t\t". 'function loadJSON() {';
$map .= "\n\t\t". 'ajax("device-gps-req.php?action=live&did='.$device.'", function(response) {'
    var JSONObject = JSON.parse(response);
    var gps_lat = JSONObject.gps_lat;
    var gps_lon = JSONObject.gps_lon;
    var haskitime = JSONObject.haskitime;
    var alarm_type = JSONObject.alarm_type;
    var temperature = JSONObject.temperature;
    if (prev_lat == gps_lat && prev_lon == gps_lon){
        return;
    }
    else{
        prev_lat = gps_lat;
        prev_lon = gps_lon;
        drawRoute(gps_lon, gps_lat, alarm_type, haskitime, temperature);
    }
});
// call ajax every 16s
$map .= "\n\t\t". 'var getNewLog = setInterval(function(){ loadJSON(); }, 16000);';

```

Figure 17. AJAX code to fetch new location data

5.9 Final functionality

The final functionality of the application was restricted to each user's type. Login, Logout and password reset functionalities were available for all active users of the system.

System administrator was allowed practically unrestricted access to all functions of the system. In addition to basic user functions, administrator could create contracts and assign them to each client. A system wide log that was restricted to administrator and that collected all user activity related information was also

created, which allowed better debugging of errors and provided convenient way to track user activity.

Client-type was restricted to management of devices and users with restriction of possible ownership and contract changes.

Basic user of the system would only need to be able to access self-related information as well as log and map information of allowed devices.

5.10 Testing

The application was tested during all four iterations, it started after the creation of the initial prototype. Testing was carried out primarily by other Presence staff that were not directly involved in the project. No specific test-cases were pre-written, instead, the testers were allowed to freely test various developed functions.

Using ready front-end CSS library such as Bootstrap was essential for allowing almost immediate testing because it provided an out-of-the-box front-end solution that met all known needs of the application.

The results provided through testing brought up several functions that seemed to be missing (e.g. Live map and password reset via email). The results also helped to clean up the clutter that was created with the amount of information shown for each device.

6 Evaluation

The primary functionality of the application was set on showing the location of the device on a map based on the location information retrieved from the database. For usability purposes, a user management system was also required.

The primary functions of the application were created and were part of the initial prototype created during the first stage of the development. The created functionality surpassed the expectations of Presence as it already included the user management and login features as well as contract creation and did not focus only on Leaflet.js integration. Because of used front-end-library, it also looked user-friendly.

Additional functionality was requested during stages three and four and included email sending upon user creation as well as password reset functionality that would provide the user with a hyperlink to reset the password. Ability to view the location of the device in 'real-time' was requested as well. All of the additional requests were delivered.

Much of the success in deliverability of the requested functionality could be attributed to the use of the selected tools and libraries. Documentation, examples as well as tutorials available for PHP, Leaflet.js and PostgreSQL were instrumental in achieving the initial prototype goals and even more so in the development of the final version of the application prototype. Using a front-end library such as Bootstrap allowed for creation of user-friendly user interface without spending tremendous time on CSS.

7 Future development and improvement

The application could be further developed to conform with the DRY programming principle to a higher degree. At this stage, this would have created complications for the future development. Over 7,500 lines of code were written out of which some 6,000 lines were divided between 6 classes and other PHP scripts. The high amount of lines can be undoubtedly attributed to relatively similar yet fundamentally different SQL queries that were used as well as intentional duplication of code for readability purposes.

The improvement in user interface and thus user experience would be likely needed as this part was not set as a priority at this stage and so it is currently entirely un-designed.

One of very obvious future development needs is making sure that the application is mobile friendly, while using Bootstrap allowed for development of a generally mobile friendly website, some parts of the application need more focusing to really fit the 'mobile-friendly' scope. The other approach would be development of an independent mobile-application for smartphones in particular.

The database structure as well as better utilization of roles and rights could also be improved once some more testing is done.

The upcoming change in the EU legislation could potentially limit the type of information that the system could legally store. To fully conform to the new legislation, Pre5ence would need to perform deeper research into what user related information they need, how that information would be stored and whether the intended use would be affected by the new legislation. This should be done before the application is further developed.

8 Discussion

This project was very interesting to work on. At times, the scope of the project seemed too big, which can likely be attributed to the fact that this project was done in a very independent fashion with minimal involvement from Pre5ence, which can be seen in both a good and a bad sense.

The lack of involvement allowed for a very unrestricted and stress-free approach, on the other hand, this required more consideration and even more self-control from me, the developer. The only clear restriction that was set at the start of the project was regarding its cost. Other restrictions were not so clear but fundamentally obvious, for example, there were already scripts written in PHP so choosing a different programming language for this project was unpractical.

Almost total freedom to develop application led to a very educational process, which allowed to experience all stages of a project from management and scheduling to actual development and testing. It also taught a great deal about PHP as well as object-oriented programming technique that was until this relatively unknown to me.

It was interesting to rediscover and re-understand the need for proper documentation as well as various diagrams (e.g. user-case) before and during the development phase of the project.

While prior work experience included very little of actual development work, I was happy to see that it allowed me to avoid possible pitfalls during the process.

It was also very satisfying to hear feedback from Presence staff after each development stage because it reinforced trust in my own skills as well as overall understanding of the project and its goals. In hindsight, I should have used more time for planning the project; this would have likely resulted in a quicker completion of the application.

The main complication of the project was to decide on the database structure and system as it was the core of the whole application. The need for different access rights and verification of various data complicated the SQL queries as well as the PHP scripting.

The second complication was the author's lack of JavaScript skills; luckily, this was relatively easy to overcome with the great documentation as well as plentiful non-official solutions for Leaflet.js.

The development of this project was supported by Web Server Programming and Cybersecurity courses that I attended from the second to the fourth stage of the development. Cybersecurity course gave more insight in the effects and dangers of SQL injection while Web server programming course taught how to avoid the issue entirely.

References

Usage of server-side programming languages for websites. Accessed on 19 February 2018. Retrieved from

https://w3techs.com/technologies/overview/programming_language/all

Usage of JavaScript for websites. Accessed on 19 February 2018. Retrieved from

<https://w3techs.com/technologies/details/cp-javascript/all/all>

Software framework. Accessed on 19 February 2018. Retrieved from

https://en.wikipedia.org/wiki/Software_framework

5 predictions on the future of the Internet of Things. Accessed on 19 February 2018.

Retrieved from <https://us.norton.com/internetsecurity-iot-5-predictions-for-the-future-of-iot.html>

Global Internet usage. Accessed on 19 February 2018. Retrieved from

https://en.wikipedia.org/wiki/Global_Internet_usage

Preſence Oy. Accessed on 19 February 2018. Retrieved from <http://www.preſence.fi>

Preſence Oy Hälyttimet [Alarm devices]. Accessed on 19 February 2018. Retrieved

from <https://www.haski.fi/>

Leaflet.js overview. Accessed on 19 February 2018. Retrieved from

<http://leafletjs.com/>

Copyright and Licence of Open Street Map. Accessed on 19 February 2018.

Retrieved from <https://www.openstreetmap.org/copyright>

Bootstrap History. Accessed on 19 February 2018. Retrieved from

<https://getbootstrap.com/docs/3.3/about/>

Top 10 Front-End Frameworks of 2016. Accessed on 19 February 2018. Retrieved

from <https://www.keycdn.com/blog/front-end-frameworks/>

HTML. Accessed on 19 February 2018. Retrieved from

<https://en.wikipedia.org/wiki/HTML>

What is CSS? . Accessed on 19 February 2018. Retrieved from

https://www.w3schools.com/css/css_intro.asp

Should You Use a PHP Framework? Five Pros and Cons. Accessed on 19 February

2018. Retrieved from <https://code.tutsplus.com/tutorials/should-you-use-a-php-framework-five-pros-and-cons--cms-28905>

Most searched for PHP frameworks. Accessed on 19 February 2018. Retrieved from

<https://trends.google.com/trends/explore?date=today%205-y&q=laravel,Symfony,%2Fm%2F0zqgdj,CakePHP,Zend>

Ajax (programming). Accessed on 19 February 2018. Retrieved from [https://en.wikipedia.org/wiki/Ajax_\(programming\)](https://en.wikipedia.org/wiki/Ajax_(programming))

Adobe Dreamweaver. Accessed on 19 February 2018. Retrieved from <https://www.adobe.com/products/dreamweaver.html>

PHP Superglobals. Accessed on 19 February 2018. Retrieved from <http://php.net/manual/en/language.variables.superglobals.php>

PHP Manual: Preface. Accessed on 19 February 2018. Retrieved from <http://php.net/manual/en/preface.php>

Use case. Accessed on 19 February 2018. Retrieved from https://en.wikipedia.org/wiki/Use_case

SQL Injection. Accessed on 15 February 2018. Retrieved from https://www.w3schools.com/sql/sql_injection.asp

SQL Injection Tutorial. 2018. Accessed on 11 March 2018. Retrieved from <https://www.w3resource.com/sql/sql-injection/sql-injection.php>

PHP prepared statements. Accessed on 11 March 2018. Retrieved from https://www.w3schools.com/php/php_mysql_prepared_statements.asp

Session hijacking attack. 2014. Accessed on 11 March 2018. Retrieved from https://www.owasp.org/index.php/Session_hijacking_attack

DB-Engines Ranking – Trend Popularity. 2018. Accessed on 11 March 2018. Retrieved from https://db-engines.com/en/ranking_trend

DBMS Comparison. 16.2.2018. Accessed on 11 March 2018. Retrieved from https://www.sql-workbench.net/dbms_comparison.html

Microsoft SQL Server vs. MongoDB vs. MySQL vs. Oracle vs. PostgreSQL. 2018. Accessed on 11 March 2018. Retrieved from <https://db-engines.com/en/system/Microsoft+SQL+Server%3BMongoDB%3BMySQL%3BOracle%3BPostgreSQL>

Agile software development. 10.2.2018. Accessed on March 11 2018. Retrieved from https://en.wikipedia.org/wiki/Agile_software_development

Tezer, O.S. 2014. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems. Digital Ocean Tutorials, 21 February 2014. Accessed on March 18 2018. Retrieved from <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>

Schwartz, B. 2009. When are you required to have a commercial MySQL license? Baron Shwartz Blog, 17 February 2009. Accessed on March 18 2018. Retrieved from <https://www.xaprb.com/blog/2009/02/17/when-are-you-required-to-have-a-commercial-mysql-license/>

Web Map Service. 2018. Accessed on March 18 2018. Retrieved from https://en.wikipedia.org/wiki/Web_Map_Service

About OGC. 2018. Accessed on March 18 2018. Retrieved from <http://www.opengeospatial.org/about>

Google Maps. 2018. Accessed on March 18 2018. Retrieved from https://en.wikipedia.org/wiki/Google_Maps#Comparable_services