Mendel University in Brno
Faculty of Business and Economics

The Creation of Web Portal For Display of Camera Stream And Meteorological Data

Bachelor Thesis

Supervisor:
Ing. Jaromír Landa, Ph.D.

Mgr. Radomír Kůs

Brno 2016

First of all, I would like to express my sincere gratitude to Ing. Jaromír Landa, Ph.D. for his supervision, excellent guidance, and constructive critique. Furthermore, my special thanks belong to Ing. Pavel Vybíral, Mgr. Jan Kučera and Mgr. Petr Černý for fruitful discussions and suggestions which helped me form this work. Last but not least, my enormous gratitude is due to all my close ones who have been exceptionally tolerant and patient with me not only during the creation of this work but throughout all my studies.

Statutory Declaration

Herewith I declare that I have written my final thesis: **The Creation of Web Portal For Display of Camera Stream And Meteorological Data**
by myself and all sources and data used are quoted in the list of references. I agree that my work will be published in accordance with Section 47b of Act No. 111/1998 Sb. On Higher Education as amended thereafter and in accordance with *the Guidelines on the Publishing of University Student Theses.*
I am aware of the fact that my thesis is subject to Act No. 121/2000 Sb., the Copyright Act and that the Mendel University in Brno is entitled to close a licence agreement and use the results of my thesis as the "School Work" under the terms of Section 60 para. 1 of the Copyright Act.
Before closing a licence agreement on the use of my thesis with another person (subject) I undertake to request for a written statement of the university that the licence agreement in question is not in conflict with the legitimate interests of the university, and undertake to pay any contribution, if eligible, to the costs associated with the creation of the thesis, up to their actual amount.

In Brno on: May 12, 2016 .....

Abstract

Kůs, R. *The Creation of Web Portal For Display of Camera Stream And Meteorological Data.* [Bachelor thesis.] Brno: Mendel University in Brno, 2016.

The thesis deals with the creation of a small-scale web portal ordered on demand by a client who had purchased a panoramic camera and a meteostation. The work firstly discusses web development technologies with a special focus on open source, free of charge solutions. Next, the client's requirements are introduced and a solution concept is presented. The concept is subsequently implemented utilizing OpenWrt shell scripting to store data from both the devices into a MySQL database and deploying a PHP-based content management system, MODX, to retrieve the data and display them onto the web portal.

Keywords

Web portal, MODX, content management system, open source technologies.

Abstrakt

Kůs, R. *Tvorba webového portálu pro zobrazení snímků z kamery a meteorologických údajů.* [Bakalářská práce.] Brno: Mendelova univerzita v Brně, 2016.

Tato práce se zabývá vytvořením drobného webového portálu objednaného klientem, jenž vlastní panoramatickou kameru a meteostanici. Práce nejprve pojednává o webových technologiích se zvláštním zaměřením na bezplatná řešení spadající pod licenci otevřeného software. Dále jsou uvedeny požadavky klienta a je představen návrh řešení. Návrh je následně implementován za použití OpenWrt shellového skriptování sloužícího k uložení dat z obou zařízení do MySQL databáze a s využitím MODX, systému pro správu obsahu, umožňujícího pomocí PHP zobrazit uložené údaje na webovém portálu.

Klíčová slova

Webový portál, MODX, systém pro správu obsahu, open source technologie.

Contents

1 Introduction

Since the prehistoric era, it is believed that the most precious article of all is knowledge. It stems from information that people interpret from their surroundings, giving them an advantage crucial for their survival or solely bringing them enjoyment. This belief has got even more significant in the modern world where survival mechanisms and behaviour have shifted from natural instincts to voluntary decisions. People yearn for knowledge to base their business success on, they crave information in their everyday lives and seek for faster, more effective and more convenient means of communication and social interaction. Thus, the Internet has become popular as the predominant medium of communication and information exchange, interconnecting billions of devices worldwide.

Although at times of the Internet inception its usage was limited mainly to universities, military, and early business systems (Castells, 2003, Chapter 1), it has blossomed into a meeting point of all agents of interaction, including ordinary people. Consequently, the amount of information on the Internet has started to grow rapidly, making it difficult for not so well-versed users to properly evaluate the reliability of an information source, often leading to unfortunate misinterpretations of the truth (Fitzgerald, 1997). Additionally, unlike institutions, companies and entrepreneurs employing IT[1] experts, individuals commonly confuse the Internet with the World Wide Web (Berners-Lee et al., 1994) as they do not find it necessary to understand subtle differences in IT terminology.

Mostly, they perceive the Internet (with all its facets) solely as a vehicle to draw someone's attention or to express themselves, or simply share something they like with others, regardless technological specifications being used. Thus, they oftentimes search for tools facilitating uploading and sharing information on the Internet with the least possible effort and previous knowledge (Ryan & Rao, 2008). As the technologies have become more advanced, cheaper, and easily accessible, those tools have advanced enough to enable users not only to share some information but to administrate their own websites and web portals.

No matter how big a step forward web technologies have made, the complexity of creation of such portals often exceeds the capabilities of common users. Therefore, it not seldom happens that more complicated parts of the portals are created on demand, i.e. outsourced, with simple administration and content management left on the shoulders of users themselves (Daniel et al., 2011). The popularity of such an approach has grown since the users avoid spending valuable time over learning technologies they do not know and rather concentrate on enriching the World Wide Web content with a particular set of information while still having power over functional aspects of their work.

This thesis concentrates on web technologies serving as tools for the creation of websites or web portals with a special focus on the technologies giving common users the best experience in term of their operability, their look, feel and touch sensation,

[1] IT – Information technology.

and, last but not least, in terms of their price. An analysis providing a brief overview of such tools follows in the next section. To support the theory with a practical example, the rest of the thesis focuses on the creation of an on-demand web portal required by a client belonging to the common users sort of customers as described above. First, the client's intentions and reasons to create the portal are introduced. The ensuing requirements are further discussed and some of them also extended during the phase of implementation. Subsequently, the pitfalls encountered during the creation of the web portal are discussed along with an evaluation of achieving the client's needs. The thesis is concluded by a comment on the future outlook of the portal.

2 Creation of Small-Scale Web Portals – An Analysis

By its definition (BusinessDictionary, April 23, 2016), a web portal is a publicly accessible doorway on the Internet to a place with a specific field of interest providing a broad array of resources services such as e-mail, chat, forums, search engines, links to other sites, personalized content, etc. Typically they are categorized into vertical and horizontal ones in relation to services they offer.

However, the definition is hardly applicable to its full extent to small-scale web portals such as those being administrated by individuals rather than companies. Most of the time, those portals are still a place aimed at users with a specific field of interest but they do not offer as many features as larger portals. In contrast, common websites are coming with more features than they used to, resulting in many of them meeting the definition. Thus, the boundaries between a web portal and a website are becoming blurry, especially on a small-scale level (Sowards, 1999).

Be it as it may, the definition is not important to common users. They seek for functioning, reliable and effective sources of information no matter how they are called[2]. Regarding individuals as web portal owners and administrators, it is also very important that the portal is easily manageable with low costs of running and maintenance.

Nearly nobody nowadays run their own servers in order to make their small-scale portals come alive. Public web hosting services are utilized instead (Chen et al., 2005), providing not only a publicly accessible place on the World Wide Web but also offering other services such as data back-up, keeping technologies up-to-date and usually a user-friendly environment to administrate the hosting along with its databases and to manage other Internet services such as e-mails and FTP[3]. Thus, many troubles going hand in hand with running a web portal are taken care of with reasonable fees.

The manageability of the content, however, stays in hands of portal's owners and administrators. That oftentimes raises unease as people have interesting information at their disposal but they do not know how to put it on a hosting they purchased. They search for administration and content management interfaces with neat front ends to manipulate everything from as they are accustomed to from other modern applications (Ryan & Rao, 2008).

Naturally, such tools already exist. However, most of the time they must be slightly modified in order to suit particular demands. For instance, the environment itself must be installed, set and linked up to other tools necessary for an undisrupted running of the portal. Moreover, very often some functionality is missing and hence it must be added, i.e. created with a programming language. That is why individuals intending to run a web portal put themselves in a position of clients for programmers and web developers who take the necessary steps instead of them. Such a creation is

[2]Thus, for the purposes of this text, the terms web portal and website could be used interchangeably.

[3]FTP – File Transfer Protocol.

a complex process since it requires a precise communication of requirements between the two sides (Khalid, 2000). On the other hand, when clients are capable of passing their intentions onto IT experts, the resulting web portals precisely fit the needs of clients and as such are worth the extra costs stemming from the client-developer cooperation.

The rest of this section provides an overview of technologies serving for the creation of web portals, ranging from the very fundamental ones to those providing easy administration and great end user experience. As the number of such tools is countless, the overview is not exhaustive. Instead, it concentrates on free-licensed tools especially useful for small-scale web portals.

2.1 Fundamental Technologies

Some web technologies are more important than others. For instance, HTML[4] is a cornerstone technology to any website as it describes its semantical structure. However, in order to run a web portal, other technologies are essential as well (Suehring & Valade, 2013).

2.1.1 HTML & CSS

As already stated, the very essence of web technologies is the HTML language as browsers can interpret the meaning of its tags and thus render HTML files to a form easily readable by end users. The language allows not only for text, it does also delineate tags for images, links to other pages and much more.

According to a survey (W[3]Techs$_{(b)}$, April 24, 2016) a pure HTML is employed on 70% of websites, leaving 30% to its extensible variant, XHTML. The newest HTML version, HTML5, means a huge progress as it moves HTML closer to modern technologies. It natively supports multimedia content and provides more semantic tags so that web browsers and search engines can understand the content more clearly. Apart from old attributes such as `id`, `class`, etc., it also brings into light new attributes such as the `data-*` attribute allowing to embed custom data.

No matter the new capabilities of HTML, it does not provide visually engaging websites as it merely assigns meaning to its content. Thus, another fundamental language, CSS[5] with its latest standard CSS3, can be used to lay out elements on a website according to a set of rules given by the programmer. The elements are recognized by their tags, classes, ids, etc. and the rules give them their colors, fonts, shapes, etc. A more profound elaboration on both HTML5 and CSS3 can be found for instance in (Sikos, 2011).

Since the outward appearance of websites or web portals is very important, a more detailed text (see the section 2.4) will look closer under the hood of modern

[4]HTML – HyperText Markup Language
[5]CSS – Cascading Style Sheets.

technologies making the visual aspects of the information provided via the World Wide Web even more attractive.

2.1.2 Databases

As web portals provide information, it must be stored at some place. Moreover, it should be kept in an intelligent and effective way, not to mention a standardized one. That is where database management systems (DBMS), sometimes loosely referred to as databases, come useful.

In general, data in databases are organized as collections of interrelated objects. DBMS are intended to manipulate the data and interact with users, mainly by means of their own variants of SQL[6] and additional procedural languages. The theory of databases and SQL is profoundly discussed in literature (Garcia-Molina et al., 2001, and many others) but such readings are far from the scope of this thesis.

Nowadays, the number of DBMSs is immense. The choice of any over the others is, as nicely depicted by the so-called no free lunch theorem, very difficult and it is dependent on an application. Whereas SQLite wins on a large scale, MySQL prevails as the leading DBMS in the field of web applications (DigitalOcean, April 23, 2016).

MySQL is an open source multi-platform software. For some functionality, it does not comply with the full SQL standard. Nevertheless, its community is large enough not to be worried about those differences. Also, a plethora of literature (Tahaghoghi & Williams, 2006, etc.) describes MySQL in fine detail. For web applications, it is widely used in LAMP[7] software stack. Thus, it (and the data stored inside) can be easily manipulated with the PHP programming language.

2.1.3 PHP

In order to build a functioning web portal, a programming language must be used since it serves as an instructor telling the portal what to do in what situation. According to a survey (W^{3}Techs$_{(a)}$, April 24, 2016), over 82% of websites currently employ PHP as their server-side language. Moreover, PHP is a keystone of many frameworks. A text dedicated to frameworks will follow in the next section (2.2) but a brief introduction to PHP itself should be stated beforehand.

PHP is an open source multi-platform scripting language which can be embedded into HTML code. It has been captured in many books (Sklar & Trachtenberg, 2003, etc.) but its best source is the PHP official manual itself (PHP Documentation Group, April 24, 2016). It includes eight data types, several control structures, exception handling and other constructs one expects from a modern programming language. Since its 5$^{\text{th}}$ version it provides improved support for object-oriented programming.

[6] SQL – Structured Query Language.
[7] LAMP – Linux, Apache, MySQL, Perl/PHP/Python.

As a consequence, the PDO[8] extension is included in the core of PHP 5 and higher distributions. The extension serves as an abstraction layer to access data from any type of database without changing the code when shifting from one to another, provided a database-specific PDO driver (e.g. `PDO_MYSQL`) is deployed. It natively supports prepared statements which are resilient against SQL injection.

Moreover, the PHP language contains the so-called superglobals which are built-in variables available in all scopes throughout a script. For instance, if one wishes to hand over some data or parameters to a script via a HTTP request, `$_POST` or `$_FILES` are capable of retrieving such data.

PHP also allows getting HTTP Cookies via the `$_COOKIE` superglobal. Cookies are small pieces of information designed to remember user preferences and actions. Since they are stored in users' browsers, they must abide by the so-called cookie law based on an opt-in principle as stated in the EU[9] directive (Directive 2002/58/EC, Article 5(3)).

2.2 PHP Frameworks

In terms of software architecture, a widely adopted pattern employed in web development is the model-view-controller (MVC) architecture. It divides the whole application into three interconnected components taking care of data itself, displaying content to users and providing communication between the components respectively. Thus, if one the components should be changed, the rest would stay untouched. Such an approach enables for easier development and better sustainability.

As the trend of programming has moved to modularity, sustainability, and extensibility, many web frameworks have emerged. Not only they enforce the MVC architecture, they also do alleviate tedious repetitive tasks such as accessing a database, user authentication, security, caching, etc. by providing multiple libraries taking action for the programmers. Many frameworks also utilize templating systems to avoid code repetition. Although some of them run on the side of the client, those based on PHP are server-side.

Currently, the most popular PHP framework is Lavarel, followed by Symphony and Nette as the most successful Czech framework (SitePoint, April 24, 2016). The overview of several PHP frameworks with their respective websites is listed in Table 1.

Those frameworks are operated through code which is welcomed by a programmer who desires to have full control over what is underneath the hood of the application. By contrast, common users see this feature as a displeasing obstruction preventing them from running their own web applications, web portals, and websites. Even though the frameworks are mostly very well documented, a common user seeks for a solution providing a neat graphical user interface. Fortunately, it

[8] PDO – PHP Data Objects.
[9] EU – The European Union.

Table 1: List of the most popular PHP frameworks in personal projects. Adapted after (SitePoint, April 24, 2016).

Name	Website
Lavarel	https://laravel.com/
Symphony	https://symfony.com/
Nette	https://nette.org/
Yii	http://www.yiiframework.com/
CodeIgniter	https://www.codeigniter.com/
PHPixie	https://phpixie.com/
Zend	http://framework.zend.com/

is not necessary to create one's own anytime it is needed since the problem can be overcome by combining framework capabilities with content management tools.

2.3 Content Management System Tools

Content management system (CMS) is a tool for carrying out content management as a set of processes leading to an effective and flexible organization, categorization, and structuring of resources such as texts, images, etc. Most of the time, such tools provide a graphical environment and enable for a collaborative work of multiple users. Although CMSs designed for managing web content are specifically referred to as web content management systems, for the purposes of this thesis the more general term will be used for the former and the latter altogether. A profound elaboration on content organization and CMSs can be found for instance in (Boiko, 2005).

Unlike frameworks, CMSs are designed for publishers, editors and other professions not requiring knowledge of programming. Very often their documentations provide step-by-step manuals to achieve particular goals as the users are expected not to be familiar with the problem domain. Furthermore, they are distributed with many presets or automated templates and they allow for add-ons such as WYSIWYG[10] editors, photo galleries, etc. Thus, such systems are also suitable for small-scale web portals.

On the other hand, exempting users from necessary adjustments of the environment they would be using may lead to curtailment of its functionality. Fortunately, as time passes, some CMSs are evolving into high-layer web frameworks. Hence, the modern ones guarantee both code-driven scalability and extensibility and easy clickable graphical user interfaces for content management.

As CMSs are exploited to organize data, their content must be stored somewhere. That is why every CMS requires a database to function. That might be

[10] WYSIWYG – What You See Is What You Get.

unfortunate in terms of hosting services prices as they depend on the number of databases and their designated space. However, since a typical web portal requires a database to store data at as well, the downside is not so severe.

The most deployed CMS of all is WordPress. Nevertheless, it does not necessarily imply its superiority over others. In fact, it is very hard to choose the best CMS as their differences are subtle. Some of the most important aspects to take into account when weighing up the pros and cons are security, ease of use, documentation, community support, functionality, and available add-ons. Some of the best-ranked (Top Ten Reviews, April 24, 2016) CMSs running on PHP are provided in Table 2 along with the DBMSs they support to store their objects and users' content.

Table 2: List of the top PHP-based CMSs. Adapted after (Top Ten Reviews, April 24, 2016).

Name	Supported Database	Website
ocPortal	MySQL	http://ocportal.com/
WordPress	MySQL, MariaDB	https://wordpress.com/
Drupal	MySQL + 6 other	https://www.drupal.org/
Joomla!	MySQL + 3 other	https://www.joomla.org/
rubedo	MongoDB	http://www.rubedo-project.org/
eZ Publish	MySQL + 3 other	http://ez.no/
MODX	MySQL	https://modx.com/

2.3.1 MODX

One of the CMSs that literally evolved into a fully customizable framework is MODX. The wordplay here stems from the fact that MODX's first distribution name was Evolution whereas its current second distribution is called Revolution. MODX's official website claims that MODX offers many great features such as flexibility and extensibility, multi-server and multi-language support, advanced caching options, SEO optimization and many others.

However, what (Top Ten Reviews, April 24, 2016) finds the most attractive about MODX is its security and ubiquitous customer support with the large and active community. Also its documentation (MODX Docs, April 24, 2016) is very well elaborated, covering the needs both of the very beginners and skillful users. The downsides of MODX stressed in the review are connected to a lack of eCommerce tools such as point-of-sale systems and affiliate tracking features.

Nevertheless, for small-scale web portals its features are more than sufficient as it offers many add-ons (referred to as packages) to be installed from inside the MODX Manager. Among many, the most popular ones are:

- `TinyMCE` – a third side WYSIWYG editor (TinyMCE, April 25, 2016) fully integrated into MODX,
- `Wayfinder` – a flexible navigation builder utilized e.g. for creating menus,
- `FormIt` – a package to create advanced web forms with custom validations, spam protection, dynamic dropdown lists, etc.,
- `getResources` – serves to create lists of resources satisfying a given set of parameters,
- `Gallery` – a front-end tool to create dynamic galleries of images.

Presumably, those packages gained their popularity since they provide solutions to problems any website faces. However, from developer's point of view, some other packages should be taken into account as well. Namely, `Ace` and `CodeMirror` packages simplify coding as they introduce word completion and syntax highlighting into MODX. `MinifyX` should be regarded as a tool speeding up web pages loading time as it compresses CSS and JavaScript files. Next, `VersionX` comes handy when someone needs to see a history of changes within MODX objects (such as documents) as it keeps their older versions in the MODX database.

Another useful package is `SimpleSearch` making it very easy to include a within-website search function. In order to run a portal in multiple languages the `Babel` package should be installed.

Once speaking about packages, it is important to introduce MODX terminology. Many of the packages come in the form of Snippets which are, loosely speaking, pieces of PHP code allowing for dynamic content customization and adding some extra functionality. A typical Snippet call with several parameters is as follows

```
[[!SnippetName? &parameter1=`value1` &parameter2=`value2` ...]].
```

The parameters after a question mark are called Properties and, in general, they are not limited to Snippets. Some of them are obligatory whereas others may be optional. Their names are usually listed in the package or Snippet specification. An exclamation mark in front of a Snippet's name indicates whether the Snippet is called cached (the exclamation mark is present) or uncached.

A golden rule of MODX programming is not to include any HTML output into Snippets[11]. Hence, MODX has introduced Chunks as bits of static text or HTML code enabling to template out Snippets' output. For instance the `getResources` Snippet may be passed several so-called tpl parameters such as `tplFirst`, `tplLast`, `tplOdd`, etc. serving to template out resources discovered by the Snippet. Additionally, Chunks might have their own Properties as well, making the development very flexible.

To complete the list of MODX objects utilized for structuring websites, MODX resources must be mentioned. Each resource corresponds to a page and it might

[11] Thus separating logic of an application, portal or website from its front end.

come in several forms. The resources are laid out onto web pages accordingly to their Templates. The Templates, as their name suggests, are an abstraction of resources and they typically contain HTML tags conjointly with MODX tags pointing to specific attributes of resources such as their name, content, etc. MODX also allows creating new MODX tags via the so-called Template Variables.

As it can be seen, MODX indeed is a fully-fledged tool for creating websites and web portals. Besides, its usage is free of charge. To sum up the enumeration of MODX features, its extension to PDO will be introduced. The name of the extension is `xPDO` and it serves as an object-relational bridge between MODX objects and relational database structures. This is very important since it enables to easily create a mapping between the two. In other words, if developers intend to store data in the MODX database they are still capable of using prepared statements and getting the results of their queries in the form of PHP objects with negligible effort.

Finally, it should be stated what makes MODX appealing to the cooperation of unskilled web portal owners with programmers as described at the beginning of this section (2). Its development-friendly environment along with high flexibility and customizability helps programmers to prepare administration accounts for their clients according to their level of programming skills. In the end, when the accounts are set properly even quite complicated websites and web portals may be administrated without any knowledge of programming whatsoever.

2.4 Graphical Features And Interactivity

Up to now, means to make a web portal functional were discussed. Although the functionality and smooth running of any application are crucial, most end users of today take them as granted. In fact, they see any flaw in web functionality as a big failure. Also, end users mostly do not understand the mechanisms behind. For those reasons, they judge the portal according to its front end, i.e. they find important the look, feel and touch sensation acquired during the usage of a portal. As portals are created in order to attract end users, it is essential to enhance their visual facets as much as possible.

2.4.1 Responsive Web Design

One of the aspects of the final end user sensation that must be taken into consideration when creating a website or a web portal stems from the number of device types being used for web browsing. The days when personal computers were the only option to access a website had come to past a long time ago. More and more often, owners of tablets and mobile phones have unlimited Internet access on their devices, making them a section of the market to focus on.

Fortunately, CSS3 have come with the so-called `@media` queries[12] (Sikos, 2011). They are capable of distinguishing various devices and define different styles for each one of them. Hence, web pages might be tailored according to, e.g.:

- the device width and height,
- the device resolution,
- orientation of the device,
- width and height of the viewport[13],
- number of colors the device can display.

However, the device types are not the only variable in the equation. According to the statistics by (W3Counter, April 25, 2016), almost 52% of all websites are accessed via Chrome, followed by Safari (16%), Internet Explorer & Edge (14%), Firefox (10%) and Opera (3%)[14]. The diversity in browsers also may cause troubles as they might render the same data and interpret some commands differently. Despite the supreme effort the web developer community makes to unify web browsers services and in that way simplify lives of many website owners, the goal has yet to be achieved. Nevertheless, many tools taking care of everyday troubles that might be encountered while creating a website or a web portal have been created. Probably the most famous one is Bootstrap (Bootstrap, April 25, 2016).

Bootstrap is an open source, free of charge front-end framework simplifying development of responsive websites. Since the release of its 3rd version, it adopted a mobile-first design philosophy. Apart from other features[15], it provides a set of ready CSS templates, such as predefined buttons, the so-called hamburger menu, responsive typography, etc., which gives websites employing Bootstrap a modern appearance. It also supports the latest versions of all modern web browsers.

Moreover, it provides predefined classes for easy layout options called grid system. Loosely speaking, grid system brings website's responsivity to life and thus it substitutes media queries which are no longer necessary. For instance, assigning a `col-xs-6` class to an HTML element makes it take 6 columns out of 12 which means that the element would occupy one-half of its parent's width on extra small devices[16] and larger. Assigning a `col-md-4` class to the same element would point out the effect of a mobile-first approach as the element would still occupy one-half of its parent's width for extra small and small devices. However, for medium devices[17] and larger, the width would change to one-third (12 divided by 4) of its parent.

[12] Although `media` types already existed in CSS2.
[13] The viewport is the user's visible area of a web page.
[14] The remaining 5% belongs to other browsers.
[15] For example jQuery plugins (see the section 2.4.2).
[16] Bootstrap defines a breakpoint for extra small devices to be 768 px.
[17] The lower breakpoint for medium devices is 992 px.

2.4.2 JavaScript And Its Libraries

Another important aspect of a web portal's front-end is its interactivity. It prevents end users from getting bored while browsing through the portal so that they would not only stay on the portal but do come back as well. However, interactivity and the amount of movement on web pages should not be exaggerated as it might, in fact, have a very opposite effect than it had been intended.

Nowadays, more than 93% (W^{3}Techs$_{(c)}$, April 25, 2016) of client-side programming, supplying websites with interactive elements, is done with JavaScript as it is a language all modern web browsers understand to. It provides a huge variety of features and numerous libraries. The most employed library, with 70% coverage out of all websites (W^{3}Techs$_{(d)}$, April 25, 2016), is jQuery.

The jQuery library (jQuery, April 25, 2016) was designed to simplify the client-side scripting of HTML document traversing, animation, event handling, etc. It is free and open source. Most of its functionality comes from a heavily overloaded `jQuery` function whose alias `$` simplifies the code even more. For instance, the JavaScript code

```
1 document.getElementById('elementId').style.display = 'none';
```

could be with the jQuery syntax rewritten as a simply as

```
1 $('#elementId').hide();
```

The second most employed JavaScript library (W^{3}Techs$_{(d)}$, April 25, 2016) is the aforementioned Bootstrap. Although its functions are jQuery-dependent, they provide additional features than the previously mentioned library. Mainly, they make the elements handled by its functions more appealing to the eye.

As it is not possible to include all of innumerable JavaScript libraries and frameworks, only a few will be introduced. More precisely, instead of pure JavaScript, those mentioned below are built upon the jQuery library. Their choice is based partly on the fact that those libraries are free of charge and partly they were selected to cover all main functionalities one might need to create a fully-fledged web portal.

Magnific Popup provides one of the very essential functionalities of almost any website or web portal as it represents an elegant way of displaying images. It is a responsive lightbox supporting both single image views and galleries. Moreover, HTML elements such as text, forms, videos and maps can be popped up with the lightbox as well. More about Magnific Popup can be found on its official website (Semenov, April 25, 2016).

bxSlider is a fully responsive jQuery slider. Although its sphere of application partly overlaps with the previous lightbox, bxSlider, as its name suggests, serves mainly for creating presentations and animations. Basically, any HTML element or

set of elements (such as pieces of text, videos or any combination) may be rendered as a slide and combined into a slideshow. Its options enable easily changing modes of transitions between slides, transition easing types, defining captions and many others. A full list of options along with several examples is provided on bxSlider's official website (Wanderski, April 25, 2016).

NVD3 is a set of re-usable charts built upon D3.js library (Bostock, April 26, 2016). D3.js is a JavaScript framework allowing for efficient data-driven manipulation of DOM[18] elements such as creating HTML tables from an array of numbers, etc. In fact, the wide spectrum of functionality covered by D3.js gave the incentive to create a smaller collection of its re-usable components in order to simplify some repetitive tasks. In the case of charts, it has led to the creation of the NVD3 project.

NVD3 provides several chart types (e.g. pie chart, scatter plot, simple line, etc.) models. Provided data to be displayed are stored in JSON[19] format, each model can be called with several parameters adjusting the final appearance of the chart. For example, a line chart can be created with the `nv.models.lineChart()` function with several parameters. Whether a single or multiple lines are displayed is driven by the data. For instance, provided `sin` and `cos` contain JSON representation of $\{x, y\}$ data points of the sine function and the cosine function respectively, the data

```
{
values: sin,      // represents the array of data points
key: 'Sine Wave', // the name of the series
color: '#ff7f0e'  // optional: color of the line
},
{
values: cos,
key: 'Cosine Wave',
color: '#2ca02c'
area: true       // a filled area chart
}
```

passed to the function would create two lines. A full documentation of all charts is provided on the official website of the NDV3 project (Novus Partners, April 26, 2016).

qTip^2 is the second generation of a jQuery tooltip plugin with easy manipulation and great flexibility. The tips, represented as speech bubbles, come in many styles and positioning options and they allow for wide range of content starting with regular text all the way to complex HTML elements. Moreover, qTip^2 natively supports

[18] DOM – Document Object Model.
[19] JSON – JavaScript Object Notation.

HTML `map`s and `svg`s. All options along with demos are well documented on qTip^2's official website (Craigh, April 26, 2016).

3 Client's Requirements And Methodology

As the crucial web technologies have been introduced, the text may proceed to the part where the creation of a web portal will be presented.

It should be noted that the portal was created on demand, i.e. for a client. At the beginning, the formulation of client's requirements was quite general, almost vague. Namely, the portal was supposed to collect data from TFA 35.1095 SINUS meteostation and Arecont Vision AV12186DN web camera and display them on the web in a neat and well-composed manner. Additionally, external web-based APIs[20] should have been utilized so that the meteorological data could be completed with further information.

The reason for those requirements can be revealed by the rationale behind this work. The client had bought both of the devices independently and then wished to display the information they provide from a single place. Although both devices naturally have their own graphical outputs, the client required accessing them both at the same time from any place, thus via the Internet.

In case of the camera, such a requirement is easy to be met since when the camera is connected to the Internet, one can simply use its IP address to display its output images and also to adjust its settings. Nevertheless, despite those web camera capabilities, the data could be solely displayed and thus no history track can be kept. Furthermore, as the client wished to see web camera images and meteorological data at one place, it was needed to combine the data. This would not have been possible if one had wanted to stick to the original software being purchased with the devices.

Besides, manipulating the data from the meteostation would be inconvenient if its original software was to be utilized. Indeed, if one aims to save and analyze meteostation data, they need to transfer the data via a USB cable to a computer. On top of that, the computer must have Windows operating system installed on and be turned on 24/7, yielding additional costs not to mention heating and unsolicited noise.

Thus, to accomplish the aforementioned requirements, it was settled with the client to purchase web hosting space along with a domain, where all the data could be transferred to and managed programmatically. In particular, a web hosting by the ONEbit company (ONEbit, May 10, 2016) was chosen[21] and the domain `webcamerabrno.cz` was registered.

In order to transfer the data from the camera and meteostation to the hosting, a router was bought to connect both the devices to the Internet. The firmware of the router was replaced by OpenWrt framework (OpenWrt, April 10, 2016) enabling to tailor the operation system to one's needs by installing particular software packages and employing router-side programming techniques to suit their application purposes. The framework was also chosen because it is a lighter variant to other Linux distributions and thus, unlike them, it can be operated even on a very cheap router

[20] API – Application Programming Interface.

[21] The decision was driven mainly by a good experience with the company.

without overloading the device. After, the camera and meteostation were connected to the router using an Ethernet cable and a USB cable respectively. More detailed description of the programming on the router and the data transfer will be discussed further on in the sections 4.1.1 and 4.1.2.

Subsequently, the client requested that the web portal must be manageable in such a manner that even a user lacking knowledge of programming can change its content and settings, which follows a simple reasoning. Since it was not completely clear what would be present on the portal at the time of its creation (and even later the content may change), the client wished to avoid hiring a programmer for small adjustments of the portal content whenever it would be needed. Therefore, MODX was selected as an environment to create the portal in since, along with standard means of PHP programming, it contains a highly customizable CMS.

As by default MODX comes with MySQL database, the web portal related data were selected to be stored in MySQL database as well. A default MODX version was extended with some extra packages to improve its safety and performance. Apart from packages such as Wayfinder, CodeMirror, getResources, etc. allowing for easier manipulation with MODX resources or better code readability, VersionX was installed in order to keep different versions of documents in case one wishes to restore an older version, MinifyX to reduce server load and optimize loading speed, and TinyMCE giving administrator an opportunity to edit documents in a WYSIWYG editor.

While everything was being installed, the client came with proposals on the functionality of the portal. Namely, MODX Snippets capable of displaying actual images and actual weather and showing the history of both were requested. Nevertheless, the client did not have their design thought through and thus its final form was being shaped in discussions as the Snippets were being implemented. However, as the discussions are not important for the purposes of this text, the section on the implementation itself (see the next section 4) will be formulated as if all subtle requirements were from the head of the client. Considering that every single specification must have gained client's approval, such a formulation is justifiable.

In the end, when all aforementioned aspects of the portal were taken care of, client's account was created so that the client could start adding content. In the following text, the account will be referred to as owner's administration account or simply the administration account.

Owner's administration account (described later in the section 4.4) is a simplified version of a full-access account so that no harm to data integrity and functions can be done by an inexperienced user. It is intended solely for content management such as texts changes, selecting colours, adding documents, etc. and settings adjustments.

As a consequence of the administration account, the client will be able to adjust the contents and settings also in the future. Should other features requiring programming were to be added, a full-access account would still be provided.

Additionally, as the portal have been brought to its existence, the client may start concentrating on the behaviour of the web portal end users. Hence, other refinements to the portal may follow. However, those extra features are not within the scope of this thesis.

4 Implementation of The Web Portal

In the previous section, a coarse-level concept of the demanded web portal was introduced along with technologies to be applied, implying also programming languages necessary for the implementation. Namely, OpenWrt's shell controls actions on the router, web-based programming is carried out in PHP along with HTML, CSS and Javascript (with its libraries) and MySQL is utilized to store all data. This section provides a more detailed description of how each functionality of the portal was implemented.

Let's remind that hosting was already bought and it runs a standard version of MODX with some extra packages added. The following text is organized as follows. The next section focuses on collecting the data from all the sources (i.e. web camera, meteostation and external APIs to gather additional information) and storing them in a MySQL database. Afterwards, main MODX Snippets demanded by the client will be introduced. Penultimately, elements giving the portal its end user appearance will be discussed. Last but not least, specifics of owner's administration account along with possible configuration settings will be presented.

4.1 Database And Data Collection

Before one starts programming the portal, it must be determined what data are necessary to be stored and in what way. Thus, the first step is to design a database of the portal. As already introduced, the information to be kept includes web camera images, data collected from the meteostation and some additional meteorological information.

At this point, it should be stated what exactly is the additional meteorological information mentioned above. First, the client wishes to store a dew point for every data entry. This, however, could be calculated from the data provided by the meteostation and thus no external API is needed. Secondly, the web portal is supposed to remember times of sunset and sunrise for each day. Although this information can be, given longitude and latitude, calculated as well, the computation is more demanding and thus the information will be collected from an external API and stored in the database. Last, a moon phase should be saved so that the information from a lunar calendar can be displayed.

Although all the data are to be stored in a single database, they are mutually independent. For instance, even though the history of images and meteorological data are both to be kept, there is no connection between the two entities apart from the times they were measured at. Moreover, the interval between the measurements depends on how often it is reasonable for them to be taken. Therefore, it is crucial to realize what is the purpose of the collected data to determine the time interval between two consecutive measurements.

In case of sunrise and sunset times, the answer is very simple – it is sufficient to save the information once a day since these times are fixed for each day. Re-

garding a lunar phase, it is also reasonable to collect the information daily although a lunar phase as usually measured (i.e. in the number of days since the last new moon) changes during the day. This difference is negligible as the age in days, i.e. a continuous variable, will be categorized into several discrete phases.

Regarding web camera images, the time interval between the measurements is strongly dependable on what the camera is focused on. Taking into account that the camera has four lenses, the final image is a panoramic lookout over Brno. As the web portal is oriented towards Brno and its local weather, the view taken by the camera is partly focused on buildings and partly on the sky. Thus, the skyline does not change as dynamically as if the camera was focused on the street with traveling cars and walking pedestrians. Hence, if one wishes to track weather changes, the time interval between two consecutive measurements should range approximately from 10 minutes to 1 hour.

In order to obtain more detailed weather statistics, the meteostation measurement must be stored. In this case, the time interval could be set to a whatever value depending on the timescale of an analysis.

As a consequence of the above-mentioned, the database scheme is very simple. In fact, as the data entries are measured with different time intervals, there is neither any one-to-one dependence. Thus, each device and external APIs information will be stored into a separate table with a universal ID, serving as a primary key, and time of the measurement for each entry. The importance of saving the time is dual. Firstly, it is a bare piece of information connecting the data entry with a moment in the past. Secondly, it can associate two entries from different tables. For example, if one looks at an image of the sky at a specific point in time and wishes to know the temperature at that point as well, this association is found on the basis of the saved time. It implies the importance of the column and as such it was defined as an index (in MySQL referred to as a `KEY`) in order to make finding relevant entries faster.

For the sake of completeness, it should be stated that the camera images will not be stored in the database physically but rather be saved on the web server's FTP and the database will hold only paths to the images. This is due to a difference between the costs of a database space and an FTP space, with the latter being far cheaper.

Moreover, to clarify the number of columns stored for the meteostation, the device should be briefly described. It is composed of the main console unit, an anemometer, a rain sensor and several (potentially up to 5) external sensors measuring temperature and humidity, where each component measures several values. An ERD[22] of the database is depicted in Figure 1.

Once the database is designed and physically created on the server, data may start to flow in and out. Since the database is relational and MODX is object-oriented, an xPDO mapping was employed in order to bridge between the database

[22]ERD – Entity-Relationship Diagram.

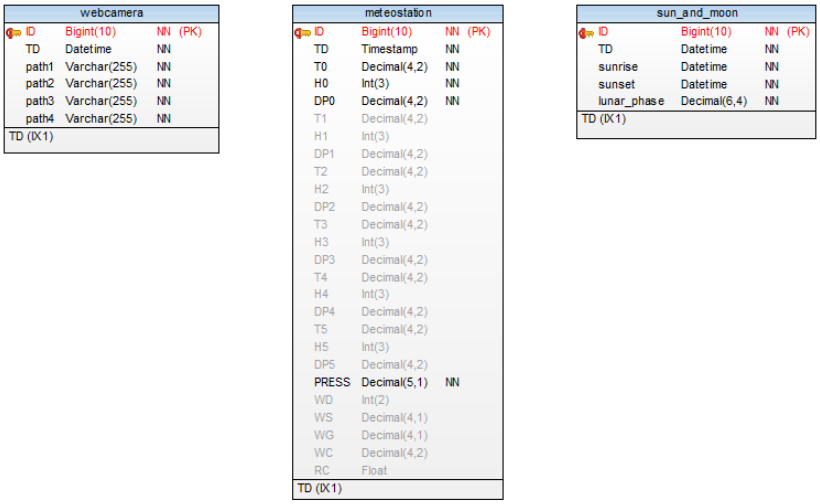

Figure 1: Entity-relationship model of the web portal's database, with no existing association among the entities. TD attributes are defined as indexes to allow for faster searching.

and MODX persistent objects so that SQL queries could be handled more easily and each database column could be accessed in PHP as an attribute of an object.

The only thing left is to find a way to transfer data from the aforementioned sources to the server. The following sections will concentrate on each source separately. At this point, it is good to recall that the web camera and meteostation data transfer will be handled by the router they are connected to instead of the devices themselves and the rest of the information will be downloaded from external APIs.

4.1.1 Web Camera

According to the manual of web camera access protocols (Arecont Vision, April 10, 2016), the images can be retrieved in several ways. One of them is through a HTTP protocol via a request in a specific format allowing for parameters such as resolution, compression ratio, etc. to be added to the call. Thus, a `wget` utility can be used to download the images to the router at any moment.

The second problem is to upload the retrieved images on the server. To do so, a `curl` utility can be used to hand over the images to a server-side PHP script in a superglobal variable $_FILES. The script must be implemented beforehand in MODX so that the images can be saved to FTP and their paths along with a timestamp stored in the database.

Since the script is easily accessible by its URL, it could be potentially triggered by anyone trying to hamper with the portal. To prevent such attempts, the script was written to run only if the IP address of a calling device matches the one of the router. The matching IP address was programmed to be adjustable in configuration settings of MODX so that the owner of the portal can change it with easy anytime.

Until now, it was explained how to send the images at a given moment. In order to upload them periodically, a shell script was programmed and put into router's crontabs so that it would run every x minutes. After some experiments, the value of x was set to be 20 which was sufficient considering the half-sky and half-buildings view of the camera. An illustrative panoramic image taken by the camera is depicted in Figure 2.

Figure 2: Panoramic image assembled from four adjacent lenses of the camera.

The downside of working with images is that they occupy quite a lot of space. More exactly, as one image takes up 256 kB of FTP space on average and given four images are sent at each moment, 3 MB are used in one hour, which adds up to around 2 GB per month. Therefore, it was arranged with the client to automatically delete images older than a given number of days from both, FTP and the database. The number of days is again adjustable in MODX configuration settings so that the owner can keep as old images as desired.

The client also wished to be able to go through the images manually and therefore they must be organized in a logical manner. Each image contains a date and time of when it was captured and a label of a lens it was taken with. The folder structure to store the images at is as follows. Each month is a folder with days as its subfolders. The folders are created on-the-fly by an auxiliary Snippet that, at the time of acceptance of new images, inserts the images accordingly to their names into their corresponding folders and if the folder does not exist, it is created. Also, when old images are to be deleted from FTP, the Snippet deletes not only the images but empty folders as well.

4.1.2 Meteostation

The idea of transfering meteostation data to the database is analogical to what was described for the camera:

- the router will periodically download the data from the device,
- a PHP script will be called by the router and the data will be passed on to the server in POST,
- if no error is encountered, the data will be saved in the database.

The meteostation is not provided with any easy way to access its data. Fortunately, as it is a problem with many meteostations, a Linux command-line utility,

which can be installed into OpenWrt, called `te923tool` (Sebastian, April 14, 2016) was created. It supports various meteostations including TFA's ones. The tool is capable of getting the data as a single line with all values separated by colons. When some of meteostation sensors are unreachable, the utility returns predefined symbols for invalid values on the corresponding positions but it still returns all the line without any interruption.

At this place, a timestamp of the download is concatenated with the line and the resulting string can be sent in $_POST with `curl` utility to a MODX script. Again, this process is inserted into router's crontabs so that it would transfer the meteostaion data every 5 minutes.

With regards to the MODX script, it must parse the incoming string into an array of values and save them to the database. Moreover, it computes dew points for every sensor with a temperature-humidity pair.

Since the meteostation is much more prone to errors than the camera, the data are checked every time they are received. The owner of the portal is informed about the errors via an email. The email address was programmed to be adjustable in MODX configuration settings along with an option to turn the notification emails completely off. Moreover, the owner can set a time period to wait before the portal sends the email. The rationale behind is that oftentimes a sensor fails several times (e.g. a WiFi connection between the sensor and a main console unit is lost for a while) and then starts working again. The skip time period is then set to filter such unsolicited notifications and let the owner know only if the sensor is down for a longer period of time (e.g. a battery is exhausted). Furthermore, since the meteostation consists of more sensors, the notifications with their skip time periods can be set for each sensor separately.

4.1.3 Additional Data Sources

Finally, information about sunrise, sunset and moon phases must be downloaded. Since neither of the devices is capable of capturing them, the data source must be external. Several APIs were tested before two of them were chosen to draw the data from. The choice was made mainly upon output simplicity so that the data traffic between the portal and an external server would be reduced to the minimum and considering whether the data are available free of charge.

The first one (Bourningsoul, April 14, 2016) offers information about the Moon such as age in days, moon phase stage, next full moon Unix timestamp, etc. in a JSON format. The call has one parameter, a Unix timestamp to get the information for, with a current timestamp as default. For example a call to `http://api.burningsoul.in/moon/1460676371` URL will result in an output as follows:

```
{
        "age": 8.161850715218,
        "illumination": 58.251584136366,
        "stage": "waxing",
        "DFCOE": 386477.57,
        "DFS": 150114497.72,
        "FM": {
                "UT": 1461302700.217,
                "DT": "5:25:00-22 Apr 2016"
        },
        "NNM": {
                "UT": 1462563078.6005,
                "DT": "19:31:18-6 May 2016"
        }
}.
```


The second one (Sunrise Sunset, April 14, 2016) also returns a simple JSON provided a latitude, longitude and date (if other than today) is attached to a call. The geographic coordinates are required as the times are dependent not only on a day of a year but also on a place on the Earth. A sample output for a `http://api.sunrise-sunset.org/json?lat=49.10&lng=16.35&date=2016-04-14` call is:


```
{
        "results": {
                "sunrise": "4:03:07 AM",
                "sunset": "5:46:12 PM",
                "solar_noon": "10:54:39 AM",
                "day_length": "13:43:05",
                "civil_twilight_begin": "3:29:45 AM",
                "civil_twilight_end": "6:19:34 PM",
                "nautical_twilight_begin": "2:48:42 AM",
                "nautical_twilight_end": "7:00:36 PM",
                "astronomical_twilight_begin": "2:03:35 AM",
                "astronomical_twilight_end": "7:45:44 PM"

        "status": "OK"
}.
```


It should be noted that the times are provided in UTC[23], thus they must be converted into a specific timezone[24] before utilized on the web portal.

[23] UTC – Coordinated Universal Time.

[24] In the case of Brno into the Europe/Prague timezone.

In general, relying upon external data sources might be troublesome as the sources might temporarily go down or might be terminated for good without a previous notice. Likewise, their licenses often claim that the volume of requests must not be excessive which is hard to abide by due to its vague definition. Therefore a MODX Snippet was designed to find a timestamp of the last entry in the database related to the `sun_and_moon` table, download JSONs from the external APIs for the next week (i.e. seven times with different timestamps) and save the received data into the database. Afterwards, the script was called four times in order to save the information in advance for an approximately one-month horizon to gain extra time if one or both of the APIs experience downtime. Subsequently, the script was set to be called automatically by the server every week so that the sunrise and sunset times along with the number of days since the last new moon would be periodically uploaded on the portal.

4.2 On-demand Snippets

As all the data are now being periodically transferred to the portal, coding of MODX Snippets to display the data stored in the database may start.

Four main functional elements were programmed and categorized into two categories:

- web camera images Snippets:
  - DisplayPhotos,
  - Animation;
- meteorological data Snippets:
  - DisplayWeather,
  - WeatherStatistics.

The following text will describe those Snippets in more detail so that by the end of this section all information needed before commenting on logical organization of web portal websites and its graphical layout will have been provided.

4.2.1 DisplayPhotos Snippet

The Snippet was created in order to display web camera images on the web portal. Its simplest call (in its uncached version), `[[!ShowPhotos]]`, displays the most recently saved images from all 4 lenses floated next to each other in order to create a panoramic image. The images are put into a wrapping `div` enabling for a Magnific Popup call giving a end user an opportunity to zoom the images in and out straight away.

On top of that, the Snippet accepts optional parameters facilitating tailoring the output to one's demands. First of all, images can be found according to a

date in `Y-m-d H:i:s` format. Secondly, a boolean parameter controls whether an extra line of text showing how old the displayed images are is added below the panoramic image or not. If the last parameter is present, instead of a panoramic image consisted of all four subimages, the output of the Snippet changes into a single image corresponding to the lens given by the parameter value.

All the parameters are summarized in Table 3. The parameters can be called in any order.

Table 3: List of DisplayPhotos Snippet parameters.

Parameter	**Description**	**Default**
&showUpdateTime	The value indicates whether information about the age of images should be displayed.	0
&updateTime	Date and time (Y-m-d H:i:s) to find images for. If the value is not present in the database, the nearest possible is selected.	*current time*
&whichLens	The number of a lens to display the images for. If present, DisplayPhotos.OneTpl chunk is used to template the out image instead of DisplayPhotos.AllTpl chunk employed in a standard call.	—

As described in the section 2.3.1, it is always a good practice to divide a back-end logic from a front-end layout. In other words, MODX Snippets should not contain HTML directly. Those pieces of HTML code should be templated out in external chunks. That is exactly what was done in this case. As the output may differ according to a set of parameters, more chunks were created in order to tailor the output precisely on what is needed. Namely, when a single image is to be displayed the DisplayPhotos.OneTpl chunk is utilized as a template whereas for panoramic images the DisplayPhotos.AllTpl chunk is utilized to distribute the images into their corresponding HTML elements.

To make a user experience with the Snippet more convenient a small extra Snippet converting a time period into `Y-m-d H:i:s` format was coded. The idea is that the user sets how many minutes, days or months the Snippet should go into the past with no need to insert the actual date. As the result, one is able to display for instance two-days old images which would change every time the Snippet is called. The extra Snippet is called OldDate and it accepts one the following parameters:

- &howManyMinutes,
- &howManyHours,
- &howManyDays.

To sum up, a toy example of the Snippet call will be presented. The resulting panoramic image of the `[[ShowPhotos? &showUpdateTime=`1` &updateTime=`[[!OldDate? &howManyHours=`6`]]`]]` call is depicted in Figure 3.

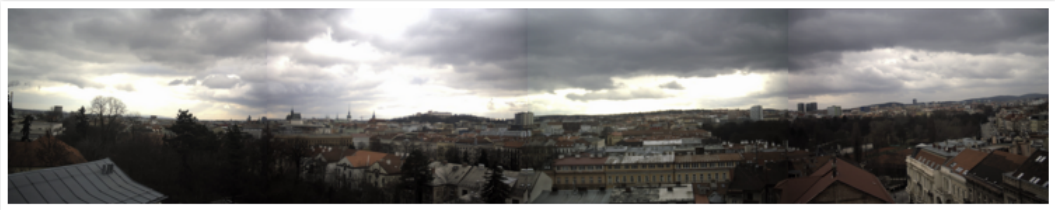

Figure 3: Panoramic image resulting from the DisplayPhotos Snippet call.

At first sight the time displayed below the image might appear confusing, but in a browser it changes dynamically as a JavaScript script is bound to the `span` wrapping the time indicator. Thus, an end user immediately recognizes which numbers correspond to hours, to minutes and which to seconds. More elaborated text about JavaScript functions and an end user experience with the web portal will be discussed further on in the section 4.3.

4.2.2 Animation Snippet

The previous Snippet was intended to display images at a given time. What if a user wanted to watch a sequence of those images to, for instance, observe how the weather has changed during last few hours or watch the Sun setting down? With DisplayPhotos the images would have to be sorted one after each other and the user would have to go through them one by one manually. Thus, another web camera related MODX Snippet, with a simple call `[[!Animation]]`, was programmed.

The idea of the Snippet is to give a user a possibility to select several images and display them as an animation. For such purposes, bxslider was chosen as an environment to display the images at since it allows for a slideshow to be set off by a simple JavaScript call with several parameters adjusting the slideshow settings.

Thus, the MODX Snippet must be capable of retrieving images from the database and organizing them into such an HTML structure so that every panoramic image will be comprehended by bxslider as one slide.

Moreover, in order to make it possible for a user to select what images to animate, a multiple `select` with times to display the images for as its options is presented. Once a user makes his choice, the images are loaded by JQuery and the animation starts. To give the end user more power over the animation, another select input corresponding to animation speed is utilized to initiate the bxslider with. For better illustration, the output of the Snippet is delineated in Figure 4.

Figure 4: End user view on the Animation Snippet with possibilities to set animation speed and select particular images to be animated.

The Snippet has a sole parameter, `&numberOfOptions`, setting how many images (or more precisely dates) would be retrieved from the database into the multiple select input.

4.2.3 DisplayWeather Snippet

`[[!DisplayWeather]]` is a Snippet call analogical to the DisplayPhotos one although the functionality is reduced (only one parameter is present). It serves for retrieving meteorological data from the database and displaying them in a desired manner. The client required a particular layout to be given to the displayed information and thus it is implicitly templated out according to ActualWeather chunk. However, if needed, other template chunks could be created. Moreover, as tracking the weather history is taken care of by another Snippet (see the section 4.2.4), no parameter altering the last stored data for some other data was implemented.

Therefore, the only parameter that could be included into the Snippet call is `&showUpdateTime` adding the time-since-data-acceptance indicator below the actual weather.

Regarding the units of measurement, the client required that the data must be available in both the imperial and the metric system units. This could be handled in PHP but JavaScript appears to be more appealing for this situation since a user might switch back and forth many times without a necessity of reloading the whole page. Thus, the MODX script inserts the data in the metric system units not only between HTML tags but as `data` attributes as well. Those values can be subsequently recomputed with a JavaScript into imperial units or redisplayed in the metric ones.

The conversion of the units, as schematically depicted in Figure 5, takes place for temperatures, wind speed[25] and for changing time formating.

[25] kt – knots.

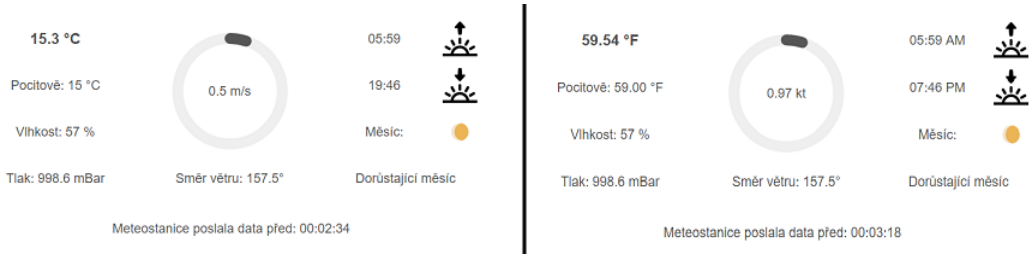

Figure 5: Comparison of the `[[!DisplayWeather? &showUpdateTime=`1`]]` call output in the metric system units (left) and the imperial units (right).

Since it is needed to remember the measure units settings across the whole website and for all end users, browser cookies are utilized to save the settings. Thus, abiding the law, a cookie alert is displayed for every user to give a consent with the cookies usage.

4.2.4 WeatherStatistics Snippet

As stated in the previous Snippet description, the WeatherStatistics Snippet, as its name suggests, was implemented to do basic statistics upon the meteorological data. To date, it has been arranged with the client that only several graphs representing the history of some particular variables would be present. Namely, the temperatures (a real temperature along with a wind chill temperature), humidity and pressure graphs will be displayed either for today or last week history. For the cases when today's graphs consist of solely a few values (especially in the mornings), yesterday statistics were added as well.

Since in the future more sophisticated graphs and statistics may follow, a JavaScript D3.js library was established as an engine to create the graphs with due to its high customizability. However as the graphs previously stated are quite simple, the NVD3 project building up upon d3.js instead of its pure form was utilized. As it expects data in a specific JSON format (see the section 2.4.2), the `[[!WeatherStatistics]]` call generates files containing those JSON strings for each graph separately as save them on FTP.

More precisely the files are created by an auxiliary Snippet and the WeatherStatistics Snippet serves as an envelope to retrieve the data from the database and call the auxiliary Snippet as many times as necessary[26]. The reason to save the data into files is the frequency of updates. As the graphs change only once in 5 minutes, it is more effective to update the files upon insertion to the database than to fetch them for every request separately.

In order to prevent long loading times, graphs related to today values, yesterday values, and last week values are stored in 3 separate files. As a default, only today statistics appear on the page. The rest is displayed only when explicitly requested

[26] In this case, it is exactly 3 times as will be mentioned in the next paragraph.

by a user triggering the `.onclick` action bound to buttons displayed on the web page. The buttons and a sample graph are shown in Figure 6.

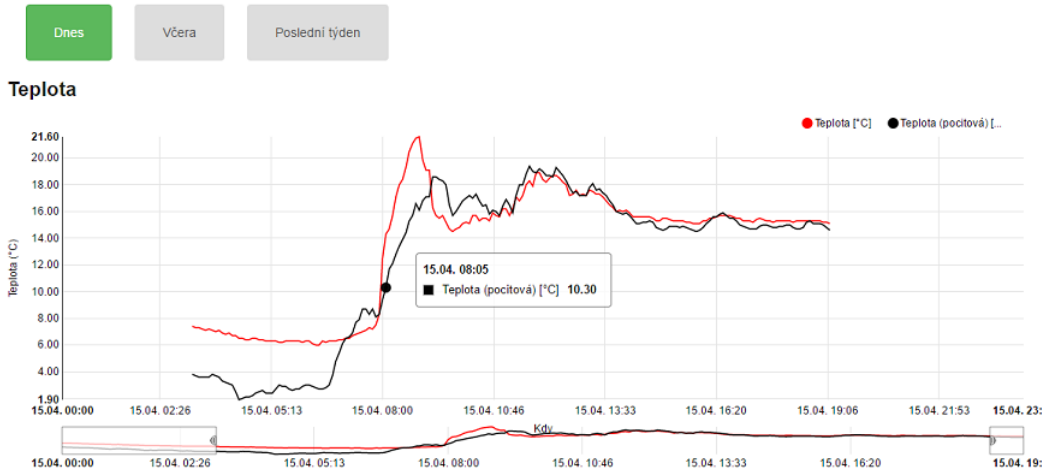

Figure 6: Sample graph of today temperatures with an option to display yesterday or last week values. The lower focus chart serves to filter the data according to x-values. When graph lines are hovered, a tick appears showing the particular values.

4.3 Organization Structure And Tweaks

Until now MODX Snippets along with some additional functionalities written mainly with JavaScript were described. They were taken out of context as stand-alone components. This section will put them all together as pieces of puzzle by explaining how they are arranged across the website. Also, some extra ideas not implemented as Snippets in MODX will be introduced. At the end of the section, graphical aspects of the website will be consulted giving the web portal's its final form.

The central element of the portal are web camera images. On that account, main website menu corresponds to web camera related functionalities. Namely, history of images, visibility, animation and landmarks of Brno are standing out as the main features. Two of them were already introduced in the sections 4.2.1 and 4.2.2 as MODX Snippets. The idea of visibility is very simple. A reference image with a good visibility is displayed simultaneously with a current image (this time a single image), output by the DisplayPhotos Snippet. A text dedicated to the last feature will be introduced later on.

Next, camera description and an about us article are present in a submenu. The submenu also contains a link to log in, possibility to switch between metric system and imperial units and in the future it will contain a choice of language[27]. The functionalities related to the meteostation are then placed in a second menu in the footer of the web portal.

[27]The English mutation of the web portal is planned but it has been currently postponed.

As the web camera is in the main focus of the portal, a current panoramic image is displayed right below the main menu straight on the homepage. Subsequently, as the meteostation with its meteorological data plays a second role in the organization structure of the web portal, the actual weather is displayed next. The actual weather block also contains links to weather statistics and information about the meteostation itself. In order to let an end user know where the camera is situated, a map with markers showing the orientation of lenses is displayed as well. For better illustration, the homepage is depicted in Figure 7.

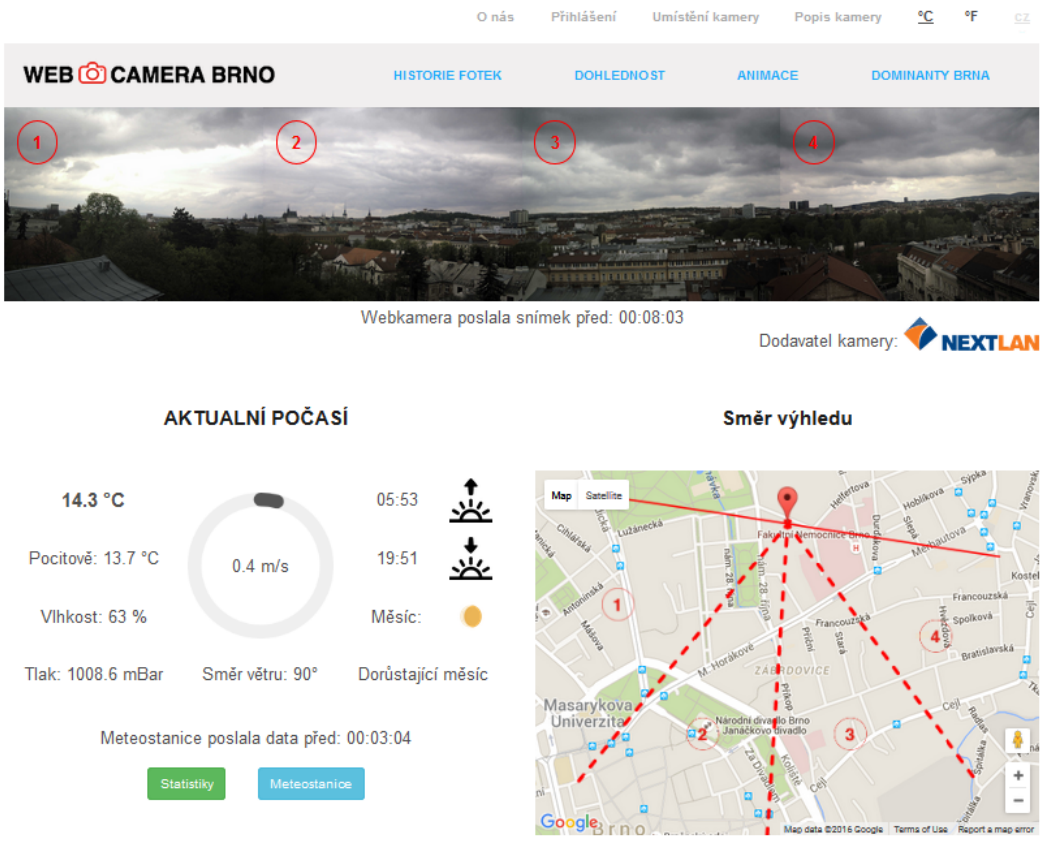

Figure 7: Cut-out of the web portal's homepage.

In order to display a map with lines and markers drawn over, a google map with a shred of extra programming was utilized. Its description does not require much attention since it is all covered by the Google Developers website (Google Developers, April 16, 2016). Brno landmarks, on the other hand, deserve their own section.

4.3.1 Landmarks of Brno

The idea of this feature is to highlight some of the buildings on a panoramic image and perform some action on them (mainly showing their description on hover).

Eventually, the feature would serve as a virtual lookout tower.

For such purposes, an HTML `map` with clickable areas can be utilized. A click on any building or place in the map can subsequently perform a whatever action such as redirecting to a website concerning the particular building, etc. Moreover, a description of the building can be displayed in a nice bubble with a qTip[2] tool. The only prerequisite is that a data attribute of each building (i.e. an `area`) is connected to the `qtip` JQuery function.

Should the feature be implemented utterly, it could serve well as a promotion of the City of Brno. However, at the moment the functionality is implemented but the content is missing since the client is not entirely sure yet what exactly to include.

The last thing left to be discussed is the web portal's responsivity. In the case of Brno landmarks a JQuery plugin, Image Map Resizer (Bradshaw, April 16, 2016) was uploaded on the portal so that the feature would be fully responsive. The next section describes how the problem of responsivity was tackled in the rest of the portal.

4.3.2 Responsive Design

As people today access websites more and more with other devices than just a PC or laptop, the design of the web portal was set to be responsive. Thus, the main representative of responsive developing, Bootstrap, was utilized to make the designing process simpler, more effective, and last but not least clearer.

Three different versions of the web portal front end were created according to Bootstrap media queries. Namely, a version for extra small devices (phones), another one for small devices (tablets) and the last one for devices with screen resolution equaled or over 992 px. The versions mainly differ in the appearance of menu and submenu with a typical hamburger menu icon appearing when the screen resolution turns out to be too small. The rest of a page is mostly adjusted by simple narrowing or widening of its elements[28].

However, some exception to this simple styling exist. Most of them are visible on the homepage as it can be seen in Figure 8. Firstly, for block elements such as the actual weather block and the map, Bootstrap grid system is utilized so that the latter wraps onto a new line on smaller devices. Columns nesting is also used as on extra small devices the middle column of the actual weather block disappears.

A big challenge was to come up with how to make panoramic images responsive. After some experiments, it was found out that adjusting their sizes makes the content of the images hardly visible. Thus, the implemented solution was to display two images instead of four in the tablet version and solely one image in the extra small one, with the least relevant images thrown away and the most interesting ones kept[29].

[28] In other words, relative units are used across the website rather than absolute ones.

[29] As the most interesting one is considered the second image since it is focused on Špilberk Castle and Petrov.

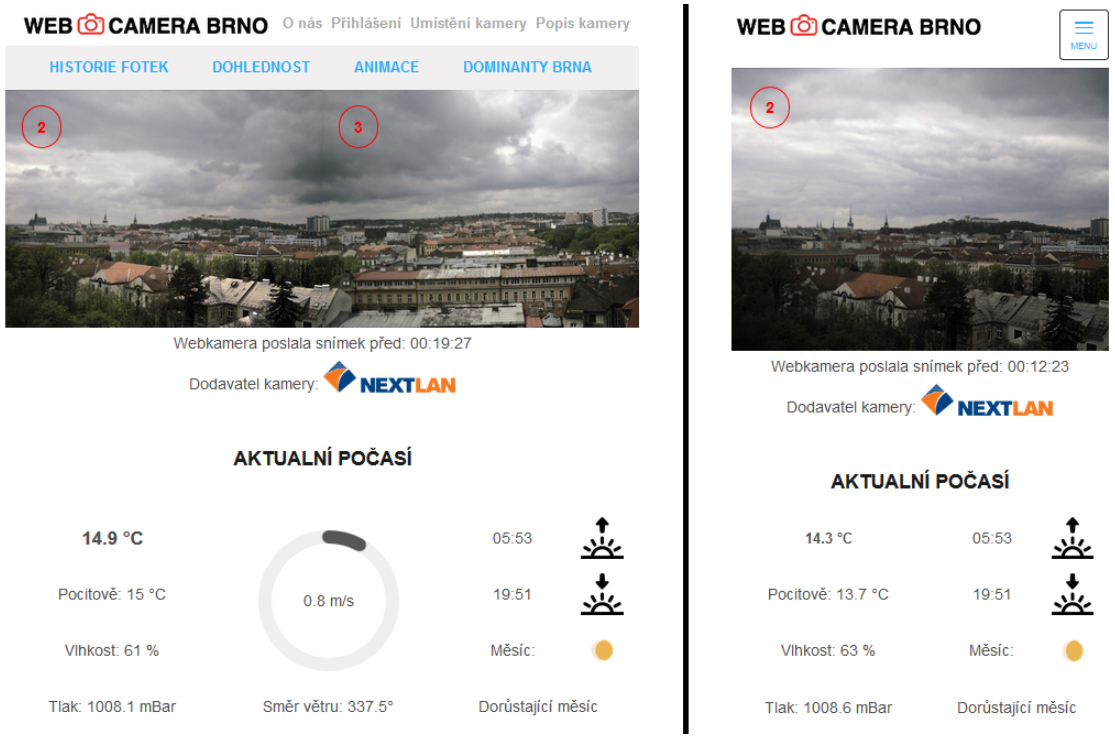

Figure 8: Tablet (left) and phone (right) versions of the web portal homepage.

4.4 Owner's Administration Account

As stated in the section 3, the owner's administration account is a stripped version of a super user MODX account. It was created so that the client can manipulate the content and adjust settings of the web portal without a necessity to call for help.

In terms of the content manipulation, the client has a restricted access to files in order not to be bothered by a MODX folder structure. In order to make documents easier to be changed, several templates and template variables were created. Moreover, as already stated the TinyMCE component was installed. An access to Snippets and chunks was prohibited for good to prevent unintentional break down of the portal.

All settings related to the client were taken out of Snippets and put into a specific category of MODX system configuration variables where they can be easily changed. Most of them such as the router's IP address, notification emails addresses and intervals, the number of days to keep images history for, etc. were covered throughout the text above. Some of the not mentioned ones include a name of the FTP directory to store camera images at, the latitude and longitude of the place where the devices are situated (for the map and sunrise/sunset estimation), a timezone and some others.

One of the last advantages of the administration account worth mentioning is a possibility to edit the content directly from the front end. More precisely, a

button displayed on the front end if the user is logged in provides a link into the manager directly to its corresponding document which is much more user-friendly than seeking the document in a MODX document tree. The difference between being and not being logged in is depicted in Figure 9. On the other hand, the edit button may obscure important elements on a page. Thus, it may be turned off and back on in the MODX configuration settings.

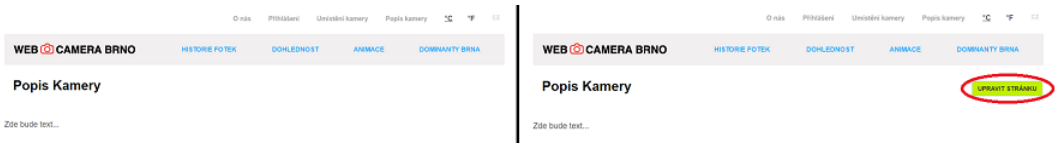

Figure 9: Comparison of the front end when accessed with a guest account (left) and the owner's administration account (right). For the latter, an edit button might be displayed.

5 Discussion And Assessment

The benefits of this work are multiple. First of all, all client's requirements were met. Moreover, some interesting ideas emerged from the collaboration. Some of them were implemented, some of them are only at the beginning and thus are not implemented utterly. For instance, the Landmarks of Brno page, serving as a virtual lookout tower over Brno, does not have a completely thought-out content yet. If implemented, it could serve as a promotion for the City of Brno.

Secondly, the work might be consulted as a guide to manipulate data from the IP camera and meteostations in general as the `te923tool` utility works for more brands. It also provides some good practices in MODX and gives an overview of several useful state-of-the-art JavaScript libraries.

The downsides or pitfalls encountered during the creation of the web portal were mainly caused by imperfections of the devices. For instance, communication between the meteostation main unit and its external sensors often fails. That was overcome by implementing sensor failures notification emails with an adjustable time interval to ignore the failures for. Also, precipitation measurements caused troubles. Some of the problems could not be overcome and thus an implementation of such features must have been dropped[30].

In regards to the camera, the deepest pitfall of all was its inability to change focal points as its lenses are rigidly bolted to the case. Unfortunately, this problem stays downgrading the sensation from otherwise delightful panoramic images. Additionally, the Animation Snippet transfers a lot of images between the portal and an end user device which may arouse suspicion of inefficient programming although the core of the problem rests with the amount of data being transferred.

5.1 Economic Evaluation

Very important is to take into account also financial aspects of the web portal. From the economic point of view, the problem being faced at the very beginning was as follows. Let's pretend that the camera and the meteostation were bought upfront. As the goal was to display data from the both, a 24/7 running computer would be needed. The costs of such a computer can not be neglected. Thus, a much cheaper[31] router was bought.

Next, in order to create and run the portal, a web hosting with a sufficient amount of FTP space and a possibility to run a database must have been bought and paid annually.

Last, the devices themselves run on electricity. The more expensive from the two is the meteostation since is consists of several sensors each having its own batteries. Nevertheless, some of the sensors may be turned off reducing not only costs for

[30] More specifically, graphs based on rain counter values did not provide reasonable values.
[31] Approximately 15 times cheaper.

those sensors but also saving batteries of the main console unit since the amount of wireless communication would be decreased.

Although some estimates could be made, it is impossible to predict the exact amount of costs per year as its final costs are dependent on:

- router's electricity consumption,
- web hosting prices,
- batteries consumption.

Nevertheless, those items are not so costly. Thus, the web portal can be viewed as very economical and, considering its convenience, the benefits of such a work to the client and possibly to society overbalance its costs.

6 Conclusion And Future Outlook

This thesis concentrated on the creation of an on-demand web portal. The client required to collect images from an IP camera and a meteostation and transfer the data somewhere to be accessible online in a nicely organized and easily adjustable manner. Those aims were achieved and even surpassed since the idea to utilize panoramic images as a virtual lookout tower over Brno emerged at the time when the concept of the portal was being formed on the basis of mutual discussion.

The fact that it is yet to be completed should not overshadow an opportunity arising from the feature. Not only can it serve as a nice functionality of the portal but also may it contribute to the City of Brno as its promotion.

However, the space to grow is much larger. In terms of increased awareness about Brno, images from the camera may be streamed to other portals about the city. Moreover, if the meteorological data were analyzed in more detail they could serve to some enthusiasts always in search for that kind of data such as hot air balloonists. For such groups of people, other dashboards providing more information on wind conditions etc. could be created.

Hand in hand with new dashboards would also probably go a creation of new access roles. Thus, other accounts than only a guest account and the administration account would be established. Besides, the owner's administration account could be utilized to display information from some hidden parts of the portal. For instance, if a few external meteostation sensors were placed inside the client's apartment they could be accessed when the client is away without being open to public. Advantages of such an access to the data span over whatever imagination might come with. For instance, the client could immediately see whether heating should be turned on or off at some point or, although the thought may sound odd, whether a neighbour should be called to water the plants. Be it as it may, the portfolio of the portal contains a multitudinous array of potential features. Hopefully, some of them will be implemented in the future enriching either the client or improving matters of public concern, or, in the best case scenario, contributing to both at the same time.

7 References

Arecont Vision. *Camera Web Page & Access Protocols* [on-line]. 2016 [last visited on April 10, 2016]. Retrieved from: http://www.arecontvision.com/supports/INTEGRATING-ARECONT-VISION-CAMERAS.

Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F. and Secret, A. The World-Wide Web. *Communications of the ACM*, August 1994, vol. 37, no. 8, p. 76–82. doi: 10.1145/179606.179671.

Boiko, B. *Content Management Bible*. 2nd ed. Indianapolis, IN: Wiley, 2005. 1176 p. ISBN 07-645-7371-3.

Bootstrap. *Bootstrap, a sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: http://getbootstrap.com/.

Bostock, M. *D3.js: Data-Driven Documents* [on-line]. 2016 [last visited on April 26, 2016]. Retrieved from: https://d3js.org/.

Bourningsoul. *MOON - API* [on-line]. 2016 [last visited on April 14, 2016]. Retrieved from: http://www.burningsoul.in/apis/moon.

Bradshaw, D.J. *Image Map Resizer* [on-line]. 2016 [last visited on April 16, 2016]. Retrieved from: https://github.com/davidjbradshaw/image-map-resizer.

BusinessDictionary. *Definition of a portal* [on-line]. 2016 [last visited on April 23, 2016]. Retrieved from: http://www.businessdictionary.com/definition/portal.html.

Castells, M. *The Internet Galaxy: Reflections on the Internet, Business, and Society*. 1st ed. Oxford: Oxford University Press, 2003. 304 p. ISBN 0-19-925577-6.

Chen Y., Das A., Qin W., Sivasubramaniam A., Wang Q. and Gautam, N. Managing server energy and operational costs in hosting centers. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '05)*. New York, NY: ACM Press, 2005. p. 303–314. doi: 10.1145/1064212.1064253.

Craigh, M.T. *qTip²: Pretty powerful tooltips* [on-line]. 2016 [last visited on April 26, 2016]. Retrieved from: http://qtip2.com/.

Daniel, F., Matera, M. and Weiss, M. Next in Mashup Development: User-Created Apps on the Web. *IT Professional*. September 2011, vol. 13, no. 5, p. 22–29. doi: 10.1109/MITP.2011.85.

DigitalOcean. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [on-line]. 2016 [last visited on April 23, 2016]. Retrieved from: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems.

Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002. *Directive on privacy and electronic communications* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved from: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:EN:HTML.

Fitzgerald, M. A. Misinformation on the internet: Applying evaluation skills to online information. *Emergency Librarian [on-line]*, February 1997, vol. 24, no. 3, p. 9–14. ISSN 03158888..

Garcia-Molina, H., Ullman, J.D. and Widom, J. *Database Systems: The Complete Book*. 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001. 1119 p. ISBN 01-303-1995-3.

Google Developers. *Google Maps JavaScript API* [on-line]. 2016 [last visited on April 16, 2016]. Retrieved from: https://developers.google.com/maps/documentation/javascript/tutorial.

jQuery. *jQuery: The Write Less, Do More, JavaScript Library* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: https://jquery.com/.

Khalid, H.M. Mass Customization and Web-Based Do-It-Yourself Product Design. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. July 2000, vol. 44, no. 12, p. 2-766–2-769. doi: 10.1145/1064212.1064253.

MODX Docs. *MODX Revolution 2.x* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved from: https://rtfm.modx.com/revolution/2.x/.

Novus Partners *NVD3: Re-usable charts for d3.js* [on-line]. 2016 [last visited on April 26, 2016]. Retrieved from: http://nvd3.org/.

ONEbit. *ONEbit hosting – domains, web hostings, servers* [on-line]. 2016 [last visited on May 10, 2016]. Retrieved from: https://www.onebit.cz/en/.

OpenWrt. *Linux distribution for embedded devices* [on-line]. 2016 [last visited on April 10, 2016]. Retrieved from: https://openwrt.org/.

PHP Documentation Group. *PHP Manual* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved from: http://php.net/manual/en/index.php.

Ryan, C. and Rao, U. Holiday users of the Internet — ease of use, functionality and novelty. *International Journal of Tourism Research*, July 2008, vol. 10, no. 4, p. 329–339. doi: 10.1002/jtr.660.

Sebastian, J. *te923tool* [on-line]. 2016 [last visited on April 14, 2016]. Retrieved from: http://te923.fukz.org/.

Semenov, D. *Magnific Popup: Responsive jQuery Lightbox Plugin* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: http://dimsemenov.com/plugins/magnific-popup/.

Sikos, L.F. *Web Standards: Mastering HTML5, CSS3, and XML*. 2nd ed. Berkely, California: Apress, 2014. 524 p. ISBN 14-842-0884-6.

SitePoint. *The Best PHP Framework for 2015: SitePoint Survey Results* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved from: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/.

Sklar, D. and Trachtenberg, A. *PHP Cookbook*. 1st ed. Sebastopol, California: O'Reilly Media, 2003. 640 p. ISBN 15-659-2681-1.

Sowards, S. Novas, Niches and Icebergs: Practical Lessons for Small-Scale Web Publishers. *The Journal of Electronic Publishing*, December 1999, vol. 5, no. 2. doi: 10.3998/3336451.0005.201. .

Suehring, S. and Valade, J. *PHP, MySQL, JavaScript & HTML5 all-in-one for dummies*. 1st ed. Hoboken, NJ: John Wiley & Sons, Inc., 2013. 720 p. ISBN 11-182-1370-X.

Sunrise Sunset. *Sunset and sunrise times API* [on-line]. 2016 [last visited on April 14, 2016]. Retrieved from: http://sunrise-sunset.org/api.

Tahaghoghi, S.M.M. and Williams, H.E. *Learning MySQL*. Sebastopol, California: O'Reilly Media, 2006. 622 p. ISBN 0-596-00864-3.

TinyMCE. *The Most Advanced WYSIWYG HTML Editor* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: https://www.tinymce.com/.

Top Ten Reviews. *The Best CMS Software of 2016* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved from: http://cms-software-review.toptenreviews.com/.

W3Counter. *Browser & Platform Market Share: March 2016* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: https://www.w3counter.com/globalstats.php?year=2016&month=3.

W^{3}Techs$_{(a)}$. Web Technology Surveys. *Usage of server-side programming languages for websites* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved

from: http://w3techs.com/technologies/overview/programming_language/all.

W^{3}TECHS$_{(b)}$. Web Technology Surveys. *Usage of markup languages for websites* [on-line]. 2016 [last visited on April 24, 2016]. Retrieved from: http://w3techs.com/technologies/overview/markup_language/all.

W^{3}TECHS$_{(c)}$. Web Technology Surveys. *Usage of client-side programming languages for websites* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: http://w3techs.com/technologies/overview/client_side_language/all.

W^{3}TECHS$_{(d)}$. Web Technology Surveys. *Usage of JavaScript libraries for websites* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: http://w3techs.com/technologies/overview/javascript_library/all.

WANDERSKI, S. *bxSlider: The Responsive jQuery Content Slider* [on-line]. 2016 [last visited on April 25, 2016]. Retrieved from: http://bxslider.com/.