FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Design of Interactive User Interfaces for Advanced Web Applications

António Filipe Magalhães Barros

Mestrado Integrado em Engenharia Informática e Computação

Supervisor at FER: Gordan Gledec (PhD)

Supervisor at FEUP: Maria Teresa Galvão Dias (PhD)

June 18, 2012

Design of Interactive User Interfaces for Advanced Web Applications

António Filipe Magalhães Barros

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor António Fernando Vasconcelos Cunha Castro Coelho
External Examiner: Doctor João Paulo Jorge Pereira
Supervisor: Doctor Maria Teresa Galvão Dias

June 18, 2012

Abstract

The web is changing, and with it, new technologies are emerging. The new Web 2.0 applications are changing the way people interact with the web pages, and the users are getting every day more demanding of the applications they use on the web.

The web is evolving fast and this evolution is helping the development of the new standards. This is not a fast change as you can see by the history and evolution of these standards, but it is what defines the path the web takes.

The aim of the project is to understand and define current trends in modern web design with respect to usability, accessibility, search optimization, performance optimization and interactivity and develop a web-based interactive application that demonstrates the key principles of user interaction on the web.

Resumo

A *web* está em mudança, e com ela, novas tecnologias estão a surgir. As novas aplicações *Web 2.0* estão a mudar a maneira como as pessoas interagem com a *web*, e os utilizadores estão a ficar cada dia mais exigentes com as aplicações que usam na *web*.

Devido a esta mudança no comportamento dos utilizadores, a *web* está a evoluir rapidamente e esta evolução está a ajudar ao desenvolvimento dos novos *standards* da *web*. Esta não é uma mudança rápida como podemos ver pela história e evolução destes *standards*, mas é o que define qual a direção que a *web* segue.

O objetivo do projeto é compreender e definir as tendências atuais do *web design* moderno em relação à usabilidade, acessibilidade, otimização de pesquisas, otimização de desempenho e interatividade e desenvolver uma aplicação *web* interativa que irá demonstrar os princípios fundamentais da interação do utilizador na *web*.

"Good design is all about making other designers feel like idiots because that idea wasn't theirs"

Frank Chimero

"Design is easy. All you do is stare at the screen until drops of blood form on your forehead"

Marty Neumier

Contents

List of Figures

LIST OF FIGURES

Abbreviations

AJAX	Asynchronous Javascript And XML
API	Application Programming Interface
CERN	Conseil Européen pour la Recherche Nucléaire (European Council for Nuclear Research)
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DOM	Document Object Model
FER	Fakultet Elektrotehnike i Računarstva
HCI	Human-Computer Interaction
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JS	JavaScript
OS	Operative System
OWL	Web Ontology Language
PHP	*PHP*: Hypertext Preprocessor
RDF	Resource Description Framework
SEO	Search Engine Optimization
SGML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocol
SPARQL	*SPARQL* Protocol and RDF Query Language
SQL	Structured Query Language
SVG	Scalable Vector Graphics
URI	Uniform Resource Identifier
URL	Uniform Resource Locators
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations
WHATWG	Web Hypertext Application Technology Working Group
WWW	*World Wide Web*
W3C	*World Wide Web* Consortium

Chapter 1

Introduction

This thesis describes the process of creating a web directory, showing the research made, explaining the implementation, doing the tests and drawing the conclusions. This is intended to do respecting the already existing standards of the web-technologies and without neglecting the user experience.

This project was developed through a collaboration between Faculdade de Engenharia da Universidade do Porto, represented by Prof. Maria Teresa Galvão Dias, and Fakultet Elektrotehnike i Računarstva — Sveučilište u Zagrebu — under the supervision of Prof. Gordan Gledec.

1.1 The Boom of The Web

Nowadays, the web keeps expanding in an unmeasurable rate. There was a boom with the massification of personal computers, but the growth continued and now expanded even more with the appearance of the smartphones, televisions connected to the internet and gaming devices.

When the *World Wide Web* appeared, it was the integration of three different specifications: HyperText Markup Language (HTML), HyperText Transfer Protocol (HTTP) and Uniform Resource Locators (URL).

The fast adoption of HTML and the slow pace of its development lead to the implementation of non-standard specifications. The browser developers seized this opportunity to add non-specified tags to their browsers, in order to take advantage in the browser wars. This had a negative impact in web development because the web developers had to choose between the support for all browsers or to provide the support for only one browser, adopting the non-standard tags that were provided by the browser.

After some time, private vendors started to provide the developers with tools to generate more appealing content, but that required the installation of external plug-ins to work.

With the standardization of the web, the aim is to walk towards a unified web, with all browsers implementing the same functionalities that are set by a board formed to keep the standards of

HTML. This board is composed by members of all the major browser developers. With HTML5, this is the path we are walking towards.

1.2 Goals

The objective of this thesis is to develop a web directory and deliver a set of guidelines on how to create a web directory using the current trends in web design. This will imply the study of several web directories, web usability and the current trends in web design, in order to be able to deliver a valid set of guidelines.

In order to test the validity of the web directory created, we need to perform some usability tests with real users. To be able to do this we need to use some measures to help us quantify and qualify the performance of the users in the tests:

1. **Ease of Learning**— how easy it is to an user who has never used the web application to learn how to accomplish basic tasks;
2. **Efficiency of Use**— how fast can the user fulfill the tasks after they have learnt how to use the web application;
3. **Memorability**— can an user who has used the web application before, use it effectively thereafter or needs to re-learn how to use it?
4. **Error Frequency and Severity**— how often the user makes errors, the seriousness of the errors and how does the system deal with the errors;
5. **Satisfaction**— how much the user likes to use the system;

1.3 Structure

This thesis is organized into three different parts, each of them relating to a different part of the development of the final solution:

Part 1 - State of the art: — This first part is the literature review about the technologies used, usability and web design.

- In chapter 2 is given a brief insight on the history of the web and some of its most used technologies.
- In chapter 3 is explained the importance of following the standards and about the guidelines in web design.

- In chapter 4 are explained the technologies used in the development of the project in a more detailed way.

Part 2 - Requirements and Implementation: — The second part is about the whole process of requirements elicitation and implementation of the project.

- In chapter 5 are specified the methodologies followed during the whole project in a concise way and is given an explanation about the results of the specification phase.
- In chapter 6 the implementation is detailed and is given the explanation about the decisions needed to make during this phase.

Part 3 - Results and Conclusions: — In this part, is made the evaluation of the whole project through usability tests performed by real users, explain the results, finalizing with the results of these.

- In chapter 7 is explained the usability tests done and show the information that was gathered.
- In chapter 8 are drawn the conclusions, taking into account all that was learned during the development phase and the tests with real users that were explained in the previous chapter.

Chapter 2

The History of Web

Even if nowadays it is hard to imagine a website without styles, it was how it worked worked in the beginning of the nineties, when the *World Wide Web* was born. Figure 2.1 illustrates the creation and evolution of HTML and Cascading Style Sheets (CSS) in a better way.

In this chapter will be given a little insight about the history of the Web and related technologies: HTML, CSS and JavaScript.

2.1 The History of HTML

The creation of HTML was on 1989, when a physicist called Tim Berners-Lee proposed a simple hypertext system, with the aim of connecting all the distributed work of his fellow physics researchers. This was integrated with two more specifications to create the *World Wide Web*: the URL and the HTTP. With these three specifications, by 1990, Berners-Lee had created the first web browser: the *WorldWideWeb* browser (figure 2.2).

He worked with is fellow researcher Mike Sendall to develop what is now seen as the first version of HTML. To create it, they followed the basics of another markup language that was promulgated as an International Standard in 1986: Standard Generalized Markup Language (SGML) [Ber93]. This markup language is still regarded as the ancestor for HTML and eXtensible Markup Language (XML) languages.

In 1992, Tim Berners-Lee published the first public HTML specification draft, with eighteen different tags [Ber92]. It's the beginning of the World Wide Web.

In order for his proposal to succeed, Berners-Lee knew that it should have the following characteristics in order for the system to be futureproof [Ber89, Vee00]:

- **Simplicity** — HTML can't be as complex as its ancestor.
- **Universality** — HTML has to be readable by any kind of computer and regardless of the Operative System.

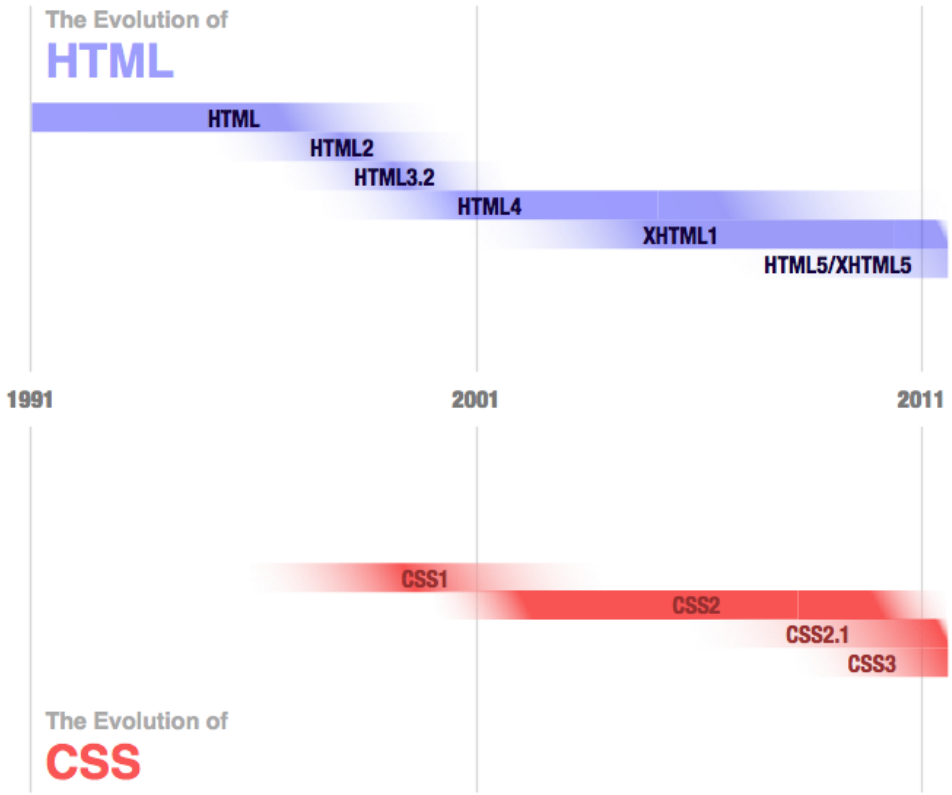


Figure 2.1: Evolution of HTML and CSS [Tea10]

- **Degradability** — any new version of the language must not break the previous versions. In other words: a new version has to offer support for the previously defined tags.

In 1994 Berners-Lee founded World Wide Web Consortium (W3C), where, together with members of some of the biggest Information Technology (IT) companies in the world, he continued to develop HTML. Amongst these companies we can find Adobe Systems, Apple, Conseil Européen pour la Recherche Nucléaire (CERN), Facebook, Google, Mozilla Foundation, Opera Software, Oracle Corporation, PayPal and Yahoo! [W3Ce].

In the second version only a few set of tags were added [Ber95], leaving this and its previous version with the same problem: it was only a structural language, with no support for page styling. This was one of the features that users were craving the most: a way to "personalize" their documents.

While the W3C was working on another version of HTML and a way to style the documents, a problem arises: browsers like Netscape and Internet Explorer, in order to satisfy their users, started to add proprietary tags [Wis]. The problem is that the tags they were adding were not in the specification of HTML and because of that, it would matter the browser users were using to

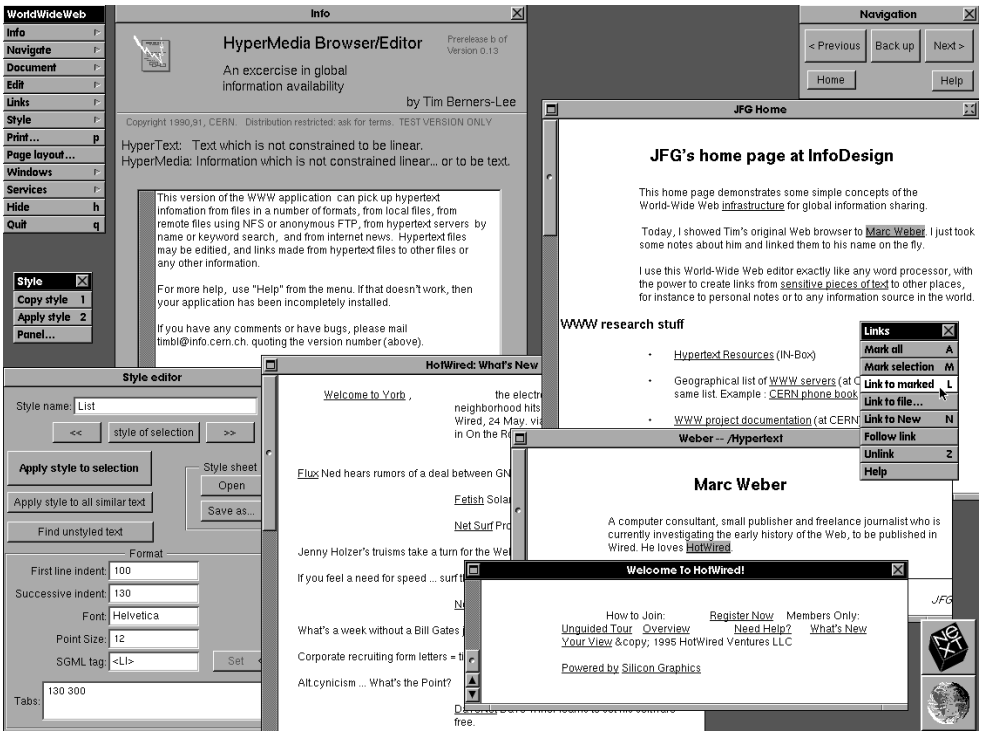


Figure 2.2: The *WorldWideWeb* browser (later renamed to Nexus to avoid confusion) was developed on a NeXT Computer [wor]

get the most out of the pages they were visiting. For example, if they visited a website made with these new Netscape tags with Internet Explorer, since Internet Explorer didn't had the support for these tags, they couldn't see it properly, and vice versa.

So, instead of dropping those proprietary tags, W3C started collecting these tags and setting them as current practices [WHA11]. This implied that the browsers should implement these "proprietary" tags so users could have the same experience, regardless the browser they were using. With this, users didn't feel the need to have more than one browser to access specific websites with proprietary tags.

At the same time, Dave Raggett introduced a new HTML draft: HTML 3.0. The changes were so big and the browsers implementations were so slow that this version of HTML had to be dropped and W3C had to turn their attention to another version of HTML: HTML 3.2 [RLAK98].

In this version W3C involved also a board made out of experts and browser companies members. This board aim was to find common ground among the various browser companies and extend the language in a way everyone agreed on. Most of the Netscape visual tags were adopted and some other proprietary tags were removed (Netscape's blink element and Internet Explorer's marquee element, for example).

In the next version they used the same strategy and it paid off. The functionalities that were added were mostly from the HTML 3.0 draft and another big improvement: support for the new presentational language: CSS.

Browser support for this version was not as troublesome as the previous versions, except for Netscape. This browser couldn't make it to this release and couldn't even handle CSS.

In the year 2000, W3C issued another version as a recommended specification: eXtensible HyperText Markup Language (XHTML). This was a new version of HTML with the strict compliance of XML language.

In 2004, and when the W3C was making efforts towards the development of XHTML 2.0, a group made of web technology fans, web browser developers and specification writers called Web Hypertext Application Technology Working Group (WHATWG) started to build HTML5 on their own. Some time after, W3C decided to drop XHTML 2.0 and work together with WHATWG in the development of HTML5 [WHA11]. It is still not a recommendation by W3C but the steps towards it are being taken. There is not a single browser that fully implements all the specifications but there are some browsers with fully implemented modules and some other modules in an advanced state of implementation [Dee10].

2.2 The History of CSS

The creation of CSS was made by another member of CERN, Håkon Wium Lie. It started when he made the proposal [Wiu94] of a scheme to style HTML documents. This was not the first version but it turned out to be a good starting point to discuss.

Bert Bos, at that time was developing Argo - a browser with style sheets already embedded in the browser, decided to join forces with Lie and, together, they submitted another proposal.

One of the ideas that Bos gave was the generalization of CSS: instead of making it only work with HTML, he recommended it to work with any type of markup language document. He took this idea from Argo, that could apply style sheets to any document [LB99].

By the end of 1996 CSS1 was declared a W3C recommendation. This was possible due to the interest of the HTML Editorial Review Board (HTML ERB), composed by representatives of IBM, Microsoft, Netscape, Novell, Softquad and W3C [RLAK98], in a way to style the documents.

In the beginning of 1997, CSS got its own space inside W3C and they started working in its next version of, with features that the first version of CSS didn't address.

The second version of CSS became a recommendation of W3C in May of 1998, and, since then, the development of any version of CSS is made by modules.

The new version of CSS, CSS3, has over fifty modules and, of those fifty, only three of them are already a recommendation by W3C - Color [W3C11a], Namespaces [W3C11b], and Selectors [W3C11c].

2.3 History of JavaScript

With the boom of the *World Wide Web*, also started the browser wars. When, in December of 1994, Marc Andreessen and Jim Clark created the Netscape Communications Corporation and alongside this created the first web browser to the masses — Netscape Navigator — they started

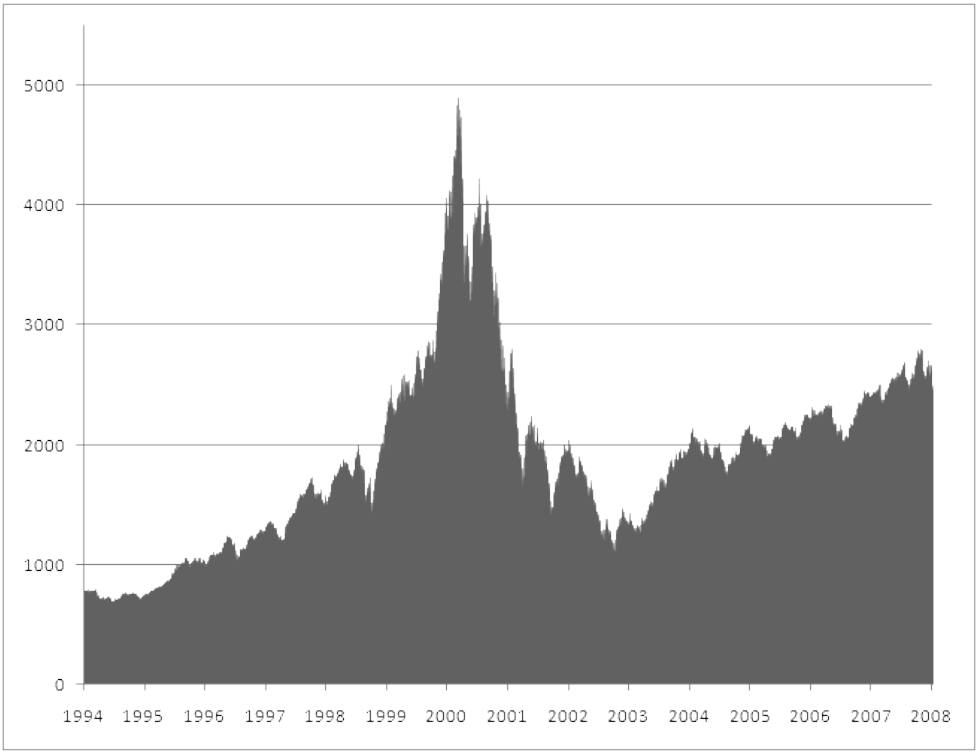


Figure 2.3: The Internet bubble timeline [Wik12]

the Internet bubble (figure 2.3). Almost overnight their browser became the most popular browser on the Web [Vee00].

In the second version of their browser they added a new feature, created by Brendan Eich, that we call nowadays JavaScript. During the development phase, it was called Mocha and after LiveScript, until they decided to name it JavaScript [You10].

Shortly after the introduction of JavaScript in the Netscape browser, they submitted it to the European Computer Manufacturers Association (ECMA) in order for it to be standardized.

In the meantime, Microsoft developed their version of JavaScript, but named it JScript in order to avoid problems of a row with Sun Microsystems over the naming rights. The first version of JScript was introduced in Internet Explorer 3.0.

In June of 1997, ECMA standardized JavaScript and created the ECMAScript standard, which embodied JavaScript core syntax. Later that year, Netscape 3.0 was released, and a new version of JavaScript was embedded: JavaScript 1.1. This new version had a few improvements, making page effects and changes on images when the mouse passed over them [Wil].

With the release of the fourth version of Netscape, another version of JavaScript was released: JavaScript 1.2. In this new version, it was possible to make changes to the CSS styles of the documents (position and visibility of elements of a page, for example), and these changes were later adopted by the second specification of CSS.

Microsoft then played their trump card: their use of a proprietary Document Object Model

(DOM) model in Internet Explorer 4.0 enabled the user to reference any part of the document thus enabling the user to have more control over the styles of the elements. Microsoft's Internet Explorer DOM model was more reliable and versatile, so W3C decided to adopt many of its syntaxes and components in their next specification of the W3C DOM model [Wil].

With the fifth version of Netscape shipped with JavaScript 1.3, but the use of the poor DOM model combined with their failure to correct the bugs that plagued their browser dictated a major change in their plans: they released their source code and stopped charging for their browser. They hoped for the open source community to take the lead and continue the development, but, instead of this, they opted to completely rewrite the core of the browser in compliance with the new W3C DOM model guidelines [Wil].

The fifth version of Internet Explorer shipped with JavaScript 1.5 and with a large set of the W3C DOM 1 recommendations and the proprietary techniques used in Internet Explorer 4. The W3C DOM recommendation, by then, allowed to create, remove and modify the elements of a page after loading, granting even more functionalities than the previous recommendations.

In 2002 a new player appears: Mozilla Firefox. Firefox was the project the open source community started when the source code of Netscape was made available. With a new rendering engine and SpiderMonkey JavaScript engine and fully implemented W3C DOM recommendation. The first steps towards a centrally recognized standard, W3C DOM, were given [Wil].

Nowadays, the W3C is on the third recommendation of W3C DOM and the last versions of Firefox, Safari, Chrome, Opera and Internet Explorer partially support this last recommendation [Wil].

Chapter 3

Guidelines for Web Design

The best way to improve the chances of getting users and keep them on a website is to have a user-centered design. Quoting Jakob Nielsen, on his book “Designing Web Usability: The Practice of Simplicity" [Nie99]:

> *The web is the ultimate customer-empowering environment. He or she who clicks the mouse gets to decide **everything**. It is so easy to go elsewhere; all the competitors in the world are but a mouseclick away.*

This, in summary, describes why the use of the guidelines are useful and extremely needed: they were made to help the developers designing a usable website that will keep the users loyal to it.

In this chapter will be given an introduction to web standards and why they are important, and will also be presented the current state of the art in both web design and web directories.

3.1 Importance of Web Standards

The web standards are a set of recommendations by W3C. These are a set of guidelines with the intent of keeping the web on the right track and to lead the Web to its full potential [W3Ca].

For the web to continue its development in order to achieve its full potential, W3C created and maintains a group of technical specifications and guidelines known to promote and help maintaining the consistency of the code which makes up a web page [W3Cd].

The advantages of correctly using and respecting these standards are many [Bre08]:

- Web pages display seamlessly in a wide variety of browsers and computers, including the new technological gadgets like Personal Digital Assistants (PDA’s) and smartphones, regardless of the operative system they use. This greatly increases the audience and the scope of the Web.

- The web standards greatly recommend the use of CSS attached to the page instead of embedding it to the page. This increases the performance because of the reduced page file size thus increasing the loading time. This also lowers the bandwidth usage for frequently visited websites.
- Increases the maintainability since this makes it easier to change all the content present in one simple page, instead of searching for the elements that need to be changed across all the pages of the website.
- The search engines are able to access and index pages more efficiently if they are according to the standards.
- As the standards are written to be compliant with the older browsers, this allows the users that are still using older versions of a browser to access correctly to the web page. This doesn't mean the user will be able to see everything that newer browsers can, but the browser will understand its structure. There are also tools to enable the developer to provide fallback solutions to older browsers.
- Developing a web site following the standards can help the developer to spend less time during the development and in the posterior maintenance phase, since it is easier to debug the code.
- Accessibility is also an important issue in web standards: this allows people with disabilities to fully use the website, but also means that people using unconventional browser to access the web, will have better support.

The Web standards that are maintained by W3C are divided in seven groups, each and every one of them responsible for several technologies/standards [W3Cd]:

- **Web Design and Applications** — this includes the standards for rendering pages, including HTML, CSS, Scalable Vector Graphics (SVG), device Application Programming Interfaces (API's) and other technologies for Web applications [W3Cg].
- **Web Architectures** — focuses on the foundation technologies and principles that support the Web, including Uniform Resource Identifiers (URI's) and HTTP [W3Cf].
- **Semantic Web** — focused on the standards of the technologies that enable people to create data stores on the Web, like Resource Description Framework (RDF), *SPARQL* Protocol and RDF Query Language (SPARQL) and Web Ontology Language (OWL) [W3Cc].
- **XML Technologies** — responsible for the standards related to XML Technologies like XML, XML Namespaces, eXtensible Stylesheet Language Transformations (XSLT), etc... [W3Cj].

- **Web of Services** — refers to message-based design like HTTP, XML and Simple Object Access Protocol (SOAP) [W3Ci].
- **Web of Devices** — focused on technologies to enable Web access anywhere, anytime, using any device [W3Ch].
- **Browsers and Authoring Tools** — responsible for the maintenance of the universality and interoperability of the Web [W3Cb].

Some of the most important standards maintained by W3C are:

- **HTML** — this is the first standard to emerge from W3C and is widely used on the web. This is one of the most used tools for designing websites.
- **CSS** — this technology allows changes in the appearance of HTML, simply declaring the styles to apply. The appearance of a complete website can be changed using CSS in a matter of seconds.
- **XML** — since XML is also a markup language, it is also based in tags, like HTML, but this is a more customizable language since it supports most of the elements that were shipped with HTML 4.0 and also allows the user to define another elements, if needed by the user.
- **XSL** — this technology is the CSS of XML: it allows changes in the way a XML document appears.
- **DOM** — this is what makes possible for a scripting language to interact with a web page. This is a language neutral interface that allows the script to dynamically access and update the content, structure and style of a web document.

3.2 Web Design

Designing is a complicated process. There's no "right" way to design a website, there are only a few useful guiding lines to help develop a usable interface. But, while these guidelines and conventions might change, human nature is a constant and the Internet did not cause its change, so far [Kru05].

That's why the best way to get good design ideas is to follow the usability engineering methodologies and try to foresee the user reactions, taking into account the user data available. It is easier to do it this way because web usability changes less rapidly than any web technology [Nie99].

Most of the times, we have the idea that every user is going to read all the text on the website, but that is not what happens: users take a glance at the website and read some text, and click on the first link that seems related to what they are searching for [Kru05].

Here are some characteristics of what we see in new web design [Nie93, Hun, Fri10, Jon10]:

- **Simplicity** — is one of the main characteristics in today's web design: a simple design captivates the user, not only by the looks but also by the usability it brings and the browsing experience it provides the user.
- **Whitespace** — this is used to make the website more clean and uncluttered, easier to read and make it appear more professional. This does not mean the space has to be white, it just means it has to be space.
- **Rapid CSS3 adaptation** — With this technology hitting the mainstream, more and more web designers are using its powers to create more visual appealing websites.
- **Strong, rich typography** — The use of new capabilities enabled by CSS3, such as "@font-face" and "text-shadow", gives us another characteristic of modern web design: the typographic revolution.
- **Soft, neutral backgrounds** — Using soft background colors give the developer the opportunity to use strong colors and, combined with the use of strong and rich typography, also provides the developer with the ability to draw the attention of the user to the most important areas.
- **Design the content, not the page** — Instead of focusing solely in designing the web page, more and more web designers choose to design its content instead. This leaves us without as many boxed layouts and, because of this, the content of the whole website looks better.

3.3 Usability

Designing the user interface of a website is complicated, so we had to apply some usability guidelines. These guidelines help us understand the users needs and gives us some guiding lines on how to design the user interface.

To help us achieve a better design, we needed to follow a set of heuristics, developed by Jakob Nielsen [Nie05]:

- **Visibility of system status** — The system should always keep users informed about what's going on, through appropriate feedback within reasonable time.

- **Match between system and the real world** — The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

- **User control and freedom** — Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

- **Consistency and standards** — Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

- **Error prevention** — Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

- **Recognition rather than recall** — Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

- **Flexibility and efficiency of use** — Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

- **Aesthetic and minimalist design** — Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

- **Help users recognize, diagnose and recover from errors** — Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- **Help and documentation** — Even though is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

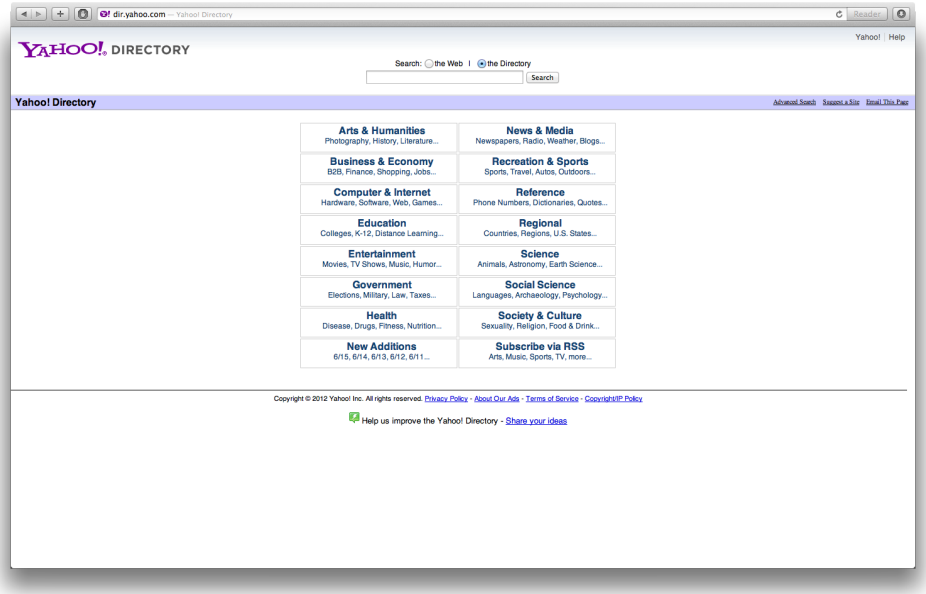


Figure 3.1: Yahoo! Web Directory [Yah12]

3.4 Web Directories

Most of the web directories that are frequently used, like Yahoo! Web Directory (figure 3.1), Google Web Directory (not maintained anymore) or the Open Directory Project Web Directory (figure 3.2), did not evolve their aspect at the same rate that the rest of the web did, keeping the same old aspect.

As pointed before, the Google Web Directory is not maintained anymore because of the change in the search paradigm: they state that searching content using a search engine is faster.

When these web directories were created, the user only needed to have the information, but nowadays, with the trivialization of the web, the users have more options than they did in the beginning, so they are more demanding than before: they want the content, but they also have high expectations on the way the content is presented.

With the failure in keeping the design of web directories according to the new standards, there is an opportunity and need to identify the user requirements and to suggest ways to address these requirements and enhance the current practices.

In general, web directories design follow the points enumerated in the previous section, but there are some other points to focus, mainly related to the way the user navigates between pages:

- **Categories and Subcategories** — this is the most common way to present the content in web directories.

- **Tag Clouds** — these are not as common as the categories and subcategories lists but, in some web directories, they are used. This enables the user to access the most commonly

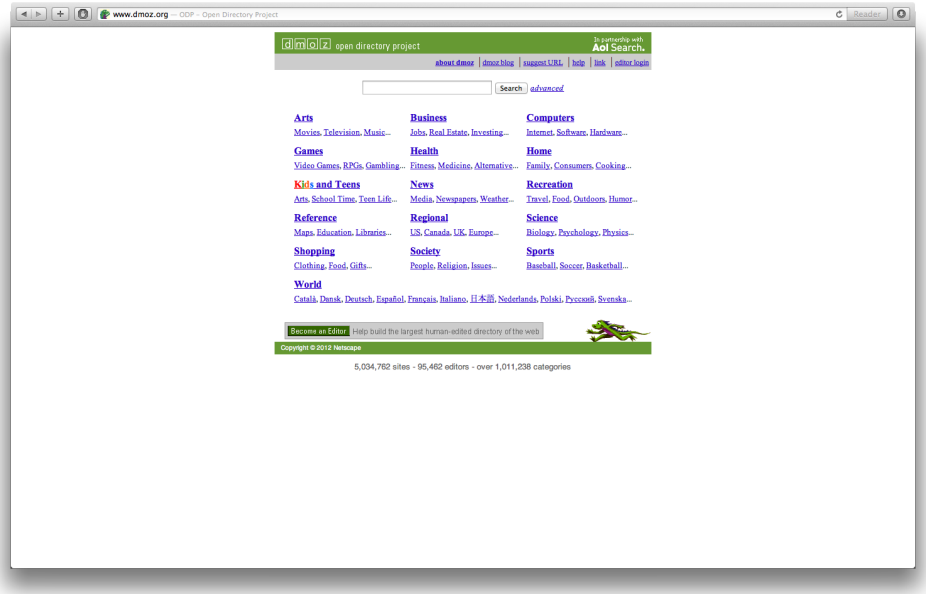


Figure 3.2: Open Directory Project Web Directory [Net12]

accessed categories and subcategories without needing to make the whole way through the list. Figure 3.3 is an example of a font-size-weighted tag cloud.

- **Search Engine** — nearly every web directory has a search engine to give the user the opportunity to make fast searches throughout the whole web directory, and, in some search engines, the ability to make searches on the whole web (through a search engine like Google). This is the fastest way to search for content since it spares the user the time to drill down the categories and subcategories to get to the results.

3.4.1 Conclusions

From the analysis of some of the most well-known web directories (figures 3.1 and 3.2), we can assert that, even if they lack the usage of new technologies, there are some important points to make it easier for the user to browse through it in an an appropriate way.

There is not much information about web design in web directories, and this leaves us space to develop a set of guidelines to help those who want to create a web directory using the new technologies that are hitting the mainstream now.

Figure 3.3: Example of a font-size-weighted tag cloud [Fri07]

Chapter 4

Modern Web Related Technologies

The technologies that are going to be used are mainly the ones that are recommended by W3C, but a little difference: it is intended to use the most advanced versions of these technologies, when possible.

Since these new versions are not fully supported by browsers, we will have to pay attention to the features added and test everything to make sure it works in the most used browsers in the market. Figure 4.1 shows us the browser usage, grouping by browser vendor (browser family).

The technologies that are going to be used are:

- **HTML** — it is intended to use the most advanced version of HTML (HTML5) when possible, but making sure it is usable by all the targeted browsers. In order to do this, it will be used a JavaScript library to indicate if the functionality works in a browser and, if it doesn't, will be provided a fallback solution to correct that.

- **CSS** — CSS is needed to apply styles to the page. It is intended to use the last version of CSS in the project (CSS3).

- **PHP: Hypertext Preprocessor (PHP)** — this will be used perform Create, Read, Update, Delete actions (CRUD) in the web server and to allow us to dynamically create web pages.

- **JavaScript** — this will be the main scripting language used in the project. JavaScript will be present in the project using a JavaScript library like jQuery or script.aculo.us.

- **Asynchronous JavaScript and XML (AJAX)** — this technology is the combination of JavaScript and CSS. This will be used to display information dynamically without the need to reload the whole page.

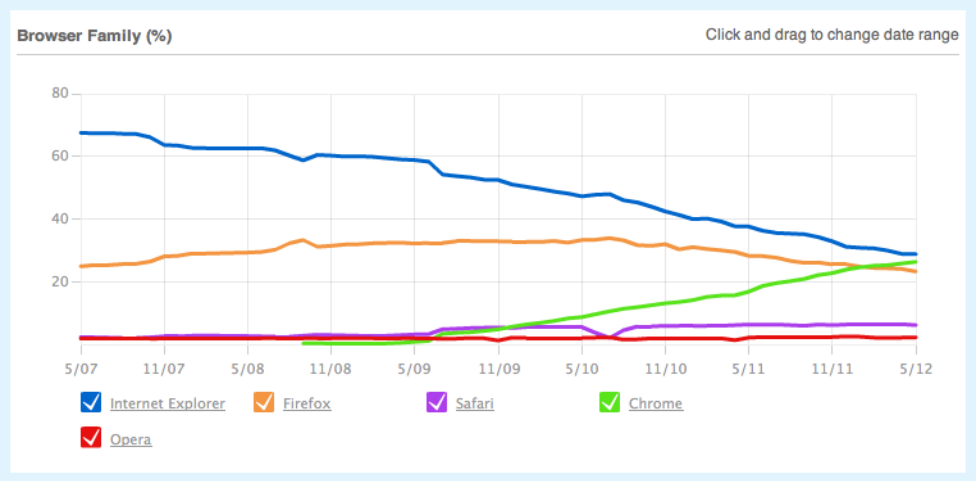


Figure 4.1: Browser usage by browser family [Awi12]

These are the technologies that are going to be used in the development of the project. Future needs might change the technologies in use, but is fair to say that these are the backbone of the application.

4.1 HTML5

HTML5 is the latest draft for the HTML specification. This version is already a Working Draft at W3C, meaning that this will be the next version of HTML. Most of the browsers are working towards this new specification, adding some of the features that are going to be present in this version of HTML.

4.1.1 Geolocation

With the non-stopping growth of the smartphones market, more and more people are using this method to access the internet from everywhere. This gives us the opportunity to combine this emergent market with the new capabilities of HTML and provide another valuable piece of information to the user: the physical place of a website.

With this information, we can provide the user information about the location of the store and, with the user location, we can show the way to go to the store in a map, including the possibility for the user to specify how he intends to go to the store (on foot, by car or by bike) and the information will be displayed according to this information.

Web Browsers		
1	Firefox 12	14.40%
2	Chrome 19	13.61%
3	Internet Explorer 9	11.39%
4	Internet Explorer 8	11.33%
5	Chrome 18	10.45%
6	Safari 5	5.40%
7	Internet Explorer 7	5.11%
8	Firefox 11	2.17%
9	Firefox 3.6	2.08%
10	iPad 5.1	1.65%

Figure 4.2: Browsers usage by browser version [Awi12]

4.2 CSS3

CSS is used to apply styles to markup languages as HTML. The specifications of CSS are maintained by W3C. CSS can be embedded in the HTML code or in a linked external file.

4.3 Scripting Languages

In order to have dynamic web pages, we need to use scripting languages to access data repositories to fill in the pages. This will be done using JavaScript and PHP. With these technologies we have covered the client side scripting and the server side scripting.

4.3.1 JavaScript

JavaScript is the scripting language that is going to be used to do all the client side programming. This will be done using two JavaScript libraries: jQuery and Modernizr.

4.3.1.1 jQuery

jQuery is a JavaScript library that exists to simplify HTML document traversing, event handling, animating and AJAX interactions, in order to have a rapid web development [The12].

4.3.1.2 Modernizr

Modernizr is a JavaScript library that enables the developers to check if the browser that is being used supports a feature of HTML5 and CSS3 and, if it doesn't, has a mechanism to help the developer provide a fallback solution, so the page doesn't loose any feature or its aspect become incoherent.

Modernizr has appeared because some of the browsers that are still being used, even if they are outdated. This is extremely useful since some of the browsers that are used are outdated and don't support some of these new features (as you can see in figure 4.2, more than 30% of the browsers currently being used are not in its newest version).

4.3.2 PHP

PHP is a server side scripting language, that enables us the dynamical creation of webpages, with information stored in a web server. Originally PHP stood for Personal Home Page Tools, but in 1997, two israeli developers, Zeev Suraski and Andi Gutmans, rebuilt the core of PHP and renamed it to PHP: Hypertext Preprocessor [NuS].

PHP allows the developers to perform CRUD actions on the web server.

4.4 Conclusion

In order to have a robust application, it needs to be well defined and that is why there are specification and definition phases, before the start of the development phase.

Before designing the web directory, this needs to be well defined to minimize the number of errors during the development phase.

In order to achieve the main objective of designing a good website we will use some of the most advanced web technologies around like HTML5 and CSS3. There may be some problems because HTML5 is still being developed but the most used browsers already have some of the new features implemented and it is the same with CSS3.

JavaScript will be used also, but preferably using a framework like jQuery or script.aculo.us, depending on the features that are going to be implemented and the features that the framework offers.

To avoid the frequent reload of the pages it will be used AJAX to address this problem while making the user experience better.

In order to validate the web application, there's the need to apply some tests to it so it could be checked for errors in the technical area and in the usability area.

The tests consist on the distribution of a list of tasks for each person, and tracking the time that they need to achieve the tasks. After that will be done a small enquiry to the user.

The application is intended to support browsers like Google Chrome, Mozilla Firefox, Opera, Safari and Internet Explorer. The user needs to be able to perform all kinds of actions supported by the application even when JavaScript is not enabled.

Chapter 5

Designing a Web Directory

The aim of this project is to develop a web directory using the modern web related technologies and, create a set of guidelines on how to create a web directory using these technologies.

In order to do this, we had to study the history of some of the most well known web technologies, like HTML, CSS and JavaScript, as well as the state of art of web technologies. This includes HTML5, CSS3 and JavaScript (using libraries and some plugins).

5.1 Specification and Methodologies

This project, as stated in chapter 1, was developed through a collaboration between Faculdade de Engenharia da Universidade do Porto and Fakultet Elektrotehnike i Računarstva.

Due to the nature of this application, we used a user centered development in all the project, adapting ourselves to the nature of the users. This method is based on the analysis and the attempt to foresee how the common user would interact with the system. However, this is not enough: for a developer, it is not easy to foresee the way the user would interact with the system, but this can be covered with tests with common users. According to Jakob Nielsen [Nie00], these kind of tests should be done with, at least, five users, but we did these tests with seven users. With this amount of users, as we can see in figure 5.1, almost 90% of the errors are covered. Normally it would be used the maximum of fifteen testers, but, because of the time restrictions, we had to make only one test, with seven testers. If we had the time, we would use the fifteen users, divided into three groups of five users, and in different phases.

During the development of the web application it was used a agile process with sprints of two weeks with meetings in the beginning of each week to discuss the work done to get feedback and to discuss the new functions for the next sprint.

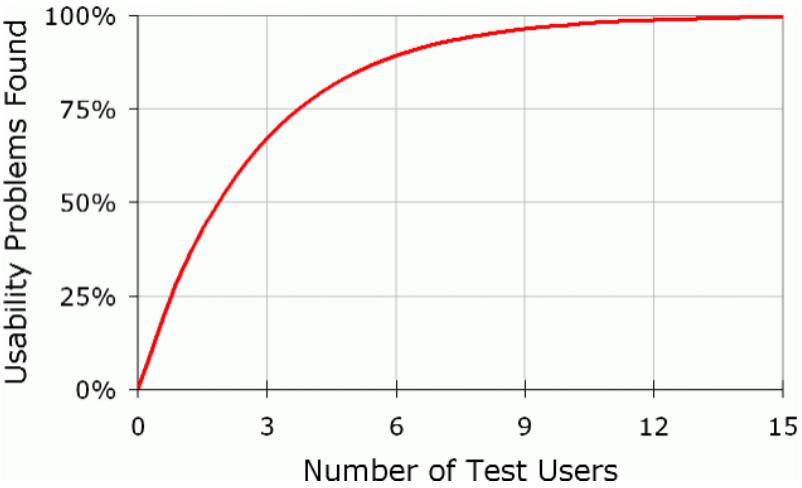


Figure 5.1: Percentage of errors found, depending on the amount of testers [Nie00]

5.2 System Requirements

Before the start of the development phase, there were discussions to elaborate the requirements of the application and we ended up with a use cases diagram 5.2, with the actions that are needed in order for the user to correctly browse through the web directory. The search engine was not supposed to be implemented, but we needed to have the action presented in the web directory, so we could gather information about the way the users would use the web directory.

Besides the actions listed in the use cases diagram 5.2, there are some more requirements we need to address while developing the web directory:

- Make use of the state of art technologies;
- Usable in a large set of browsers without any flaws;
- Web directory not to rely heavily on JavaScript;
- The web directory should be usable on smartphones;
- Accommodate different kinds of users, by providing different ways of browsing the web directory (search, tag cloud, categories and subcategories list);
- Make it easy for users to interact with the system;
- Provide feedback to the user, when executing more complex tasks;

5.3 Mockups

Before the implementation phase, we made the mockups of the pages we needed to create. This are just guiding lines on the design of the web directory. We just needed to create four different

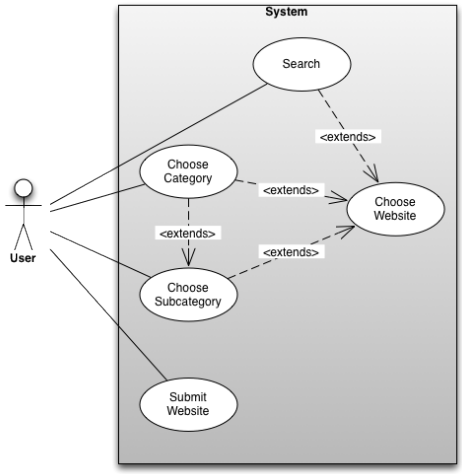


Figure 5.2: Use cases diagram with the possible user interactions with the system

mockups, since the categories page will be used for all the categories. The same applies to the subcategories. The pages we created were:

- The Home page (figure 5.3);
- Categories page (figure 5.4);
- Subcategories page (figure 5.5);
- Submit form page (figure 5.6);

These pages were just guiding lines, and they could be changed if we felt the necessity to add or change some things in the design.

5.4 Summary

In the project, we adopted a user-centered design methodology when designing the interfaces. Firstly we designed the website, taking into account the requirements and trying to put ourselves in the user's point.

In the meetings that we had before the development we decided to have a meeting every two weeks to assure the good development of the application and to be sure the schedule was being followed, and to assure that all the deliverables were correctly developed.

During these meetings, we also discussed the tests script, and the testing methodologies. With this script we want to cover every possible action the user could do within the web directory, in order for the test to be the more accurate possible.

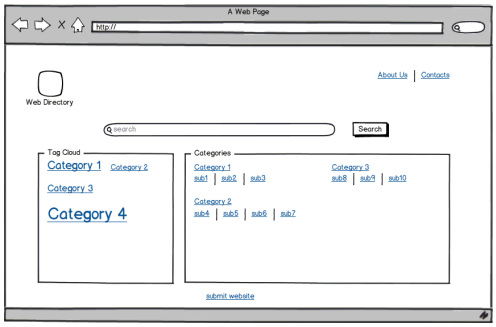


Figure 5.3: The Home page mockup

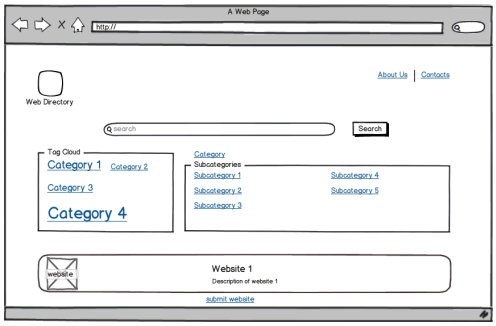


Figure 5.4: The Categories page mockup

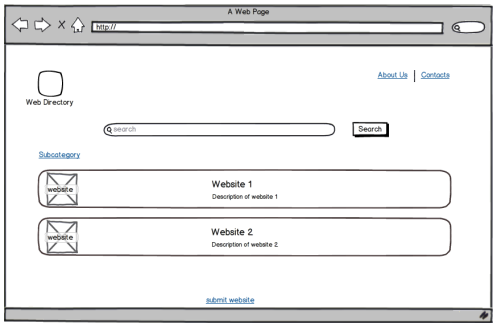


Figure 5.5: The Subcategories page mockup

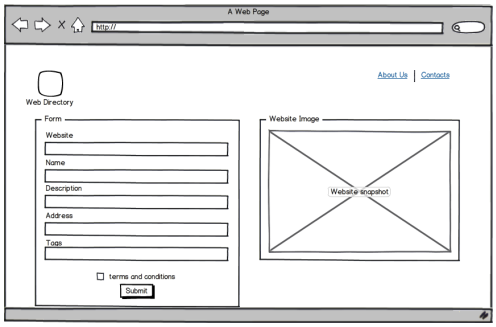


Figure 5.6: The Submit form mockup

Chapter 6

Implementation

In this chapter will be explained the architecture of the system, explaining the technologies and for what they are used and, when necessary, will be given any explanation related to the implementation decisions that were needed to take.

6.1 Architecture

The whole web application is divided into two different parts: one for the server and another for the client. This can be better understood with the aid of figure 6.1. The communications between the server and the browser are done through HTTP.

6.2 Server

The configuration of the server is a very well known for the web: MAMP. MAMP stands for Mac OS X, Apache, MySQL and PHP. The client of the application consists in a server and a database: Apache server and MySQL database; the communications between the client and the database that is running on the server is done using PHP.

We chose to use this because of its easy configuration and due to its ease of working and because of the familiarity with this configuration. The use of MAMP is due to our use of Mac OS X, but there are similar tools for Linux and Windows: LAMP and WAMP, respectively.

6.2.1 Database

To support the website was created a database 6.2 with the tables needed to insure the web directory has enough data to make the usability tests. This required the creation of five tables:

Categories — this table hosts the categories of the web directory. It contains the id, name and number of times the category was accessed.

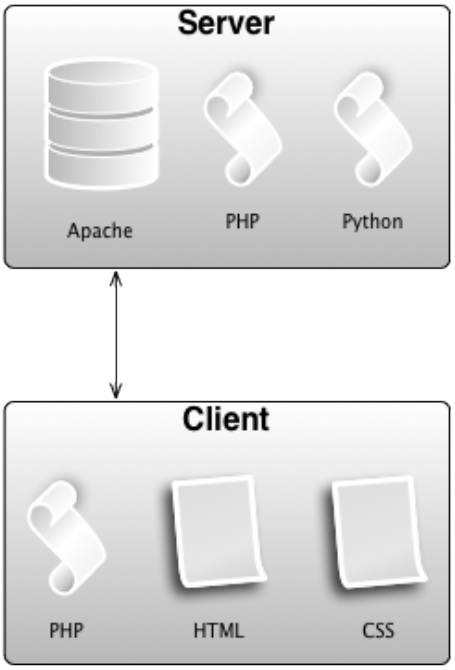


Figure 6.1: High level architecture of the system. Arrows indicate HTTP communications

Subcategories — this table hosts the subcategories of the web directory. It contains the id, name, category and number of times the subcategory was accessed.

Websites — this table hosts all the websites in the web directory. It contains the id, name, description, website, latitude and longitude and the number of times the website was accessed by the users of the web directory.

Website_Categories — this table is needed to make the connection between a website and a category. It contains the id of the website and the id of the category. This set (id of the website and id of the category) is unique.

Website_Subcategories — this table is needed to make the connection between a website and a subcategory. It contains the id of the website and the id of the category. This set (id of the website and the id of the subcategory) is unique.

6.3 Client

As one of the main requirements was to use JavaScript at its minimum due to the restrictions some people impose in order to be secured, the communications between the client and the server will be done mostly using PHP.

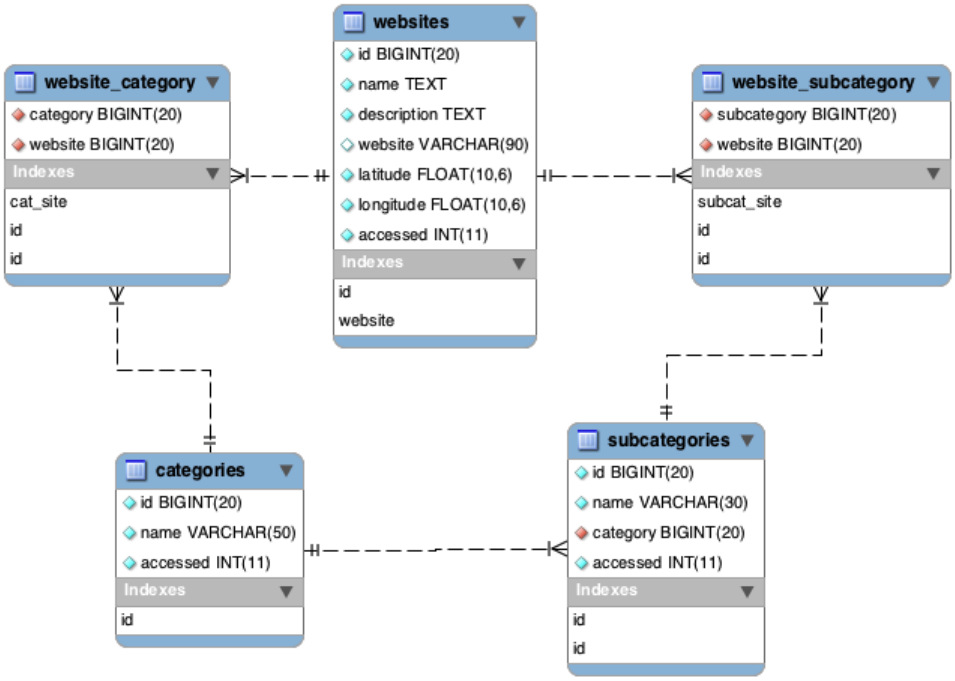


Figure 6.2: Database schema

Since every page is generated dynamically, every time the user navigates to another page, a HTTP request is made to the server and, depending on the information passed to the next page, a new page is generated.

In some pages, the use of JavaScript couldn't be avoided, but this only interferes in one of the requirements: submitting a website to the web directory.

6.3.1 Tag Cloud

As was pointed out in the chapter 3.4, one of the characteristics of a web directory is the presence of a tag cloud. With a tag cloud, the user can view the categories or subcategories and, depending on the font size, can assert the most and least accessed categories and subcategories: the bigger the font size, higher the number of times it was accessed.

This was one of the requisites of the system, so the best option was to design it in a way that it wouldn't need JavaScript so everyone could see it properly, even people browsing with JavaScript disabled.

The mechanism behind the tag cloud is simple: there are ten different levels of importance, the more important the level, the bigger the text will be. In order to calculate the levels, it was made a calculation that gives us the difference between each level, and, depending on the numbers of accesses that a category or subcategory has, it will be placed in one of those levels. The formula to calculate the step between levels is:

$$\frac{a_m - a_l}{10} = l_s \tag{6.1}$$

and to calculate the level of the subcategory:

$$ceiling(\frac{a_c}{l_s}) = l_c \tag{6.2}$$

where:

a_m is the number of accesses of the most accessed category or subcategory;

a_l is the number of accesses of the least accessed category or subcategory;

l_s is the step between each level of the tag cloud;

a_c is the number of accesses of the category or subcategory we want to calculate the level;

l_c is the level of the category or subcategory;

The greater the difference between a_m and a_l, the higher the value of l_s.

Using the following data as an example:

$a_m = 176$

$a_l = 9$

$a_c = 76$

Now applying the data listed previously to the formula in 6.1:

$$l_s = \frac{176 - 9}{10} \tag{6.3}$$

The result of 6.3 would be:

$$l_s = 16.7 \tag{6.4}$$

Now applying the result of 6.4 to 6.2

$$l_c = ceiling(\frac{76}{16.7}) \tag{6.5}$$

The result of 6.5 is:

$$l_c = 5 \tag{6.6}$$

So, with this values, the level of the category or subcategory would be five, meaning that the category wouldn't get much emphasis on the tag cloud, but it wouldn't be on the lowest levels.

When the value of l_c is higher than nine, the category level will always be ten, not having more than this number of levels.

6.3.2 Python Scripts

All the required features were implemented, but in some cases, we could not implement them using PHP or JavaScript due to the characteristics of these scripting languages.

6.3.2.1 webkit2png

As the users tend to want to see the website they are going to before really go there, we added a thumbnail of the website's home page so the user could check if the website is useful or not. This feature eases the way the users interact with the system, providing them relevant information about the websites they are searching.

In order to have the thumbnail of the website we had two different options:

User Added Image

The option of prompting the user the snapshot of the website when submitting the website could be used if we did not want to overload the server. But there are a lot of downsides with this approach: the image size could be different from what we were expecting; the image could be in a format we don't have a way to correctly process; and would be easily outdated.

Server-Side Application Snapshot

This option enabled us to, from time to time, take a snapshot of all the websites in the web directory. Using this, the user does not have to worry about uploading a picture of a website when submitting it, since we do it automatically; with this approach we can control the snapshot format and size. The downside of this approach is that if we have a lot of websites on the web directory, this can overload the server, but we can control it having a schedule to take the snapshots (for example: we can choose several sites a day and take the snapshots and, the next day, choose the websites with older images).

In a time where websites change in a blink of an eye, having an up-to-date image of the website is a necessary requirement. Without this, the web directory would be easily outdated and would lose credibility to the user: that's why we chose to use the server side application snapshot option.

This script enables the server to take snapshots of the websites in the web directory. This script uses WebKit, the rendering engine behind Apple's Safari and Google Chrome.

This could not be done in JavaScript because it is a client-side scripting language, and the images needed to be saved on the server, so we had to discard the option of doing this in JavaScript.

PHP is a server-side scripting language, but it lacks the functionalities to use the WebKit API to take the snapshots.

The possibilities were to write a new script using a language that could connect with the WebKit API or to try to find a script that could do it. We used a script developed by Paul Hammond [Ham] that fulfills our needs.

This concrete script only works on Mac OS X, but there's a version for Linux that uses libkhtml, the library of the Konqueror browser used in KDE environments.

6.3.3 PHP Scripts

As we used this scripting language to generate the dynamic pages, we tried as much as possible to use this language in other scripts, to maintain consistency throughout the project and only in cases where the language couldn't provide us the necessary conditions to use it.

6.3.3.1 Page Parsing

This script is used to parse a web page and extract information available in the meta tags of the page. The objective of this script is to spare the user the time to fill in the information of a website.

Currently it already parses the following tags:

- `<title>`
- `<meta name="description">`
- `<meta name="keywords">`
- `<meta name="geo.position">`

The content of the `<title>` tag is used to get the name of the web site: in a web directory, normally, only home pages are added, so, if the web site has this tag, we can automatically fetch it, sparing the user the time needed to add this, and the information will probably be more accurate.

The content of the `<meta name="description">` tag is used to get the description of the website, and after being fetch is added automatically to the description of the website, but, if the user wants, can be changed.

The content of the `<meta name="keywords">` tag gives is used to get the keywords of the website. These keywords currently are not used, but can be used to categorize the website. This would imply the implementation of a categorization system.

The content of the `<meta name="geo.position">` tag gives us the physical location of a website. This physical location can be a store, or the headquarters of the company that owns the website. This is used to provide the user the information about the distance from the place the user is located to the place of the website.

6.3.3.2 Image Manipulation

When searching for a website in a web directory, in order to provide better information to the users, we attach a snapshot of the websites present on the web directory.

When a user submits a website, a snapshot of the website being added is taken, as explained in section 6.3.2. This snapshot has to be manipulated to be with the correct size and to be in the format we need, in order to use it.

When adding a website to the web directory, the user has to provide the URL of the website, and a snapshot of the website is taken, to show the user a preview of the website. The problem is that the image that is displayed when adding the website is too big for what we need, so, after the website is added, the website snapshot preview is resized, to occupy less space on the server, since the thumbnail of the site that is going to be shown is a small image.

6.3.4 jQuery Plugins

As jQuery is a rich library, there are a lot of plugins already developed that add a lot of useful functionalities. This is extremely useful since these plugins normally have a good stability, are well maintained since the community contributes detecting the bugs and, sometimes, helping to solve them and another big reason to use them is to lose less time implementing features that have already being implemented.

6.3.4.1 GMAP3

This plugin was used to smooth the interaction of the web application with the Google Maps API [Goo]. The use of this is connected to the use of the Geolocation functionality of HTML5. We could provide the user the information about his distance to the website's physical location just with his position coordinates and the coordinates of the website's physical location. As the coordinates would not be easily read by the user, we converted the website's physical location to a easily understandable format: the address.

This is also used to provide the user the address when filling the information of the website. This is very useful to facilitate the task of the user, when providing the physical address of the website the user is adding to the web directory. This enables the user to input the address instead of the coordinates, since the coordinates would not be very intuitive to the user and they would be easily confused with this method.

6.3.4.2 Autocomplete

In the case of this web directory, we have a closed group of categories and subcategories, and the user does not have the possibility of adding a new category or subcategory. So, when adding a website, the user needs to be able to see the categories and subcategories the user can use. To make it easier for the user to have this information, we added this feature, that shows the user the categories and subcategories the user can choose from.

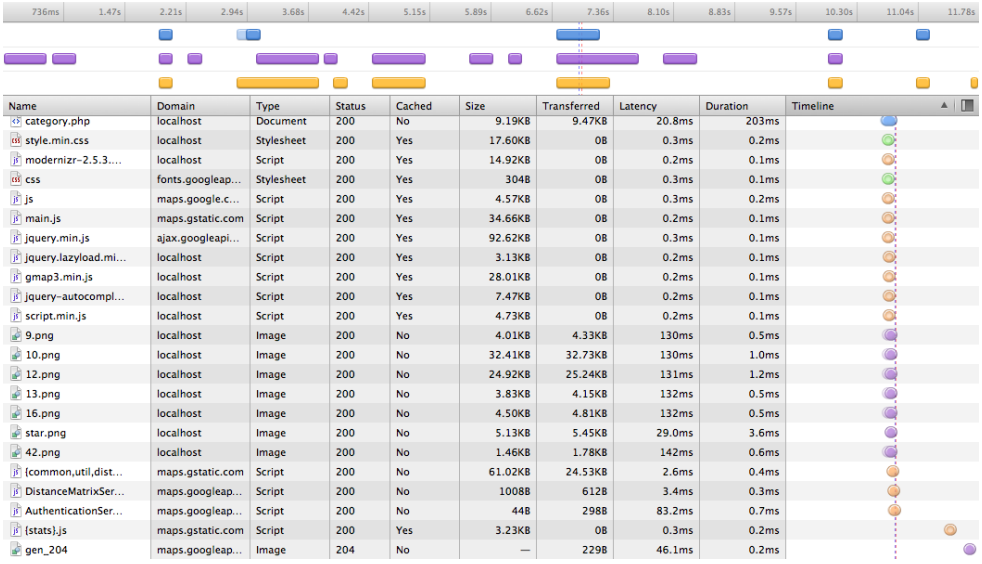


Figure 6.3: Loading time of the website not using jQuery Lazyload Plugin

As the information like the tags accepted in the categorization are all in the database and the users can't add new tags, when adding a website, the user needs to know which categories are possible and the best way to provide this information to the user is to present this when the user is adding the website.

This plugin was also used to present the information about the address of the website's physical location. Google doesn't have *all* of the addresses in their database, so it is needed to provide a way for the user to try to input the address and, if they don't find, they can ignore the address since it's an optional value, or they can choose an address that exists on Google's database and it's close enough to the website's physical location.

6.3.4.3 Lazyload

All the websites listed in the web directory will have a picture of their main page. This can have a bad impact in the loading times of the web page. To address this problem, instead of loading all the pictures at the beginning, the pictures are loaded on demand: when a user scrolls the web page, the pictures of the listed websites are fetched from the database and then the blank picture is replaced by the website picture.

This increases the performance, since the pictures are not loaded in the beginning, making the user spend less time when loading the website. For example, a website that is in the end of the list, if the user doesn't scroll until the end, its picture will not be loaded.

Basically, their pictures will not be loaded, sparing time and bandwidth to the user, as can be seen in figures 6.3 and 6.4.

In figure 6.3 we can see the images of the websites (9.png, 10.png, 12.png, 13.png, 16.png and 42.png)the thumbnails of every website of that page are loaded before the browser sends the DOM event signalizing the page is fully loaded.

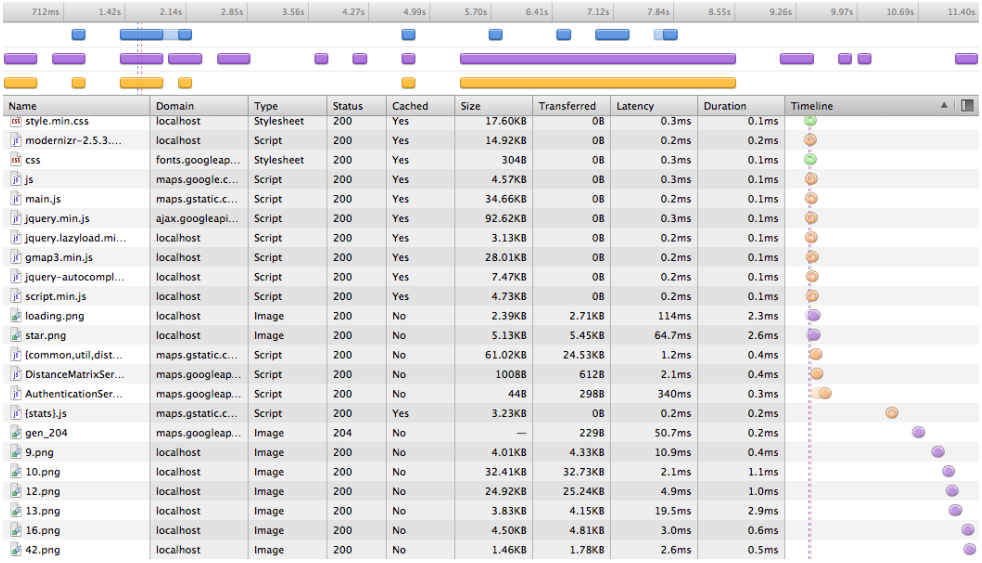


Figure 6.4: Loading time of the website using jQuery Lazyload Plugin

In figure 6.4 one thumbnail is loaded to be used in all the website thumbnails and only when the user sees a website, the image associated with that website is loaded.

As some of the users don't enable JavaScript to run, a fallback solution was provided but it will work like in the figure 6.3.

6.3.5 File Compression

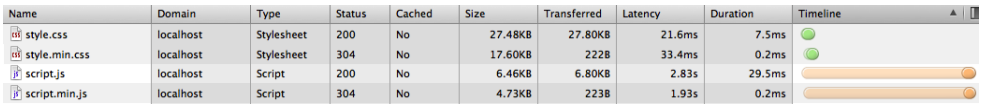
File compression reduces the loading times of the webpages. This works by applying the compression to the CSS and JavaScript files. This removes unnecessary white spaces and comments, reducing significantly the size of the files.

Applying this to the CSS and JavaScript files, help getting a better performance by reducing the initial loading times, as can be seen in figure 6.5.

6.4 Results

Resulting of the design and implementation processes, we successfully implemented the web directory. As a result, we have the interfaces of the web directory. The design of the interfaces are similar to the mockups presented in section 5.3, but there are slight changes in some of them. We also have another interface that wasn't presented in section 5.3: the categorization interface. This wasn't a requisite, but we wanted to know the opinion of the users about a new feature, so we had to implement this new interface. The interfaces we implemented were:

- The Home page (figure 6.6);
- Categories page (figure 6.7);



Name	Domain	Type	Status	Cached	Size	Transferred	Latency	Duration	Timeline
style.css	localhost	Stylesheet	200	No	27.48KB	27.80KB	21.6ms	7.5ms	
style.min.css	localhost	Stylesheet	304	No	17.60KB	222B	33.4ms	0.2ms	
script.js	localhost	Script	200	No	6.46KB	6.80KB	2.83s	29.5ms	
script.min.js	localhost	Script	304	No	4.73KB	223B	1.93s	0.2ms	

Figure 6.5: Comparison of loading times of CSS and JavaScript files with and without compression (files with ".min" are compressed)

- Subcategories page (figure 6.8);
- Submit form page (figure 6.9);
- Categorization form page (figure 6.10);

These screenshots were taken in a computer with Mac OS X Lion 1.7.4 installed and the browser was Apple Safari 5.1.7.

We also tested the application in other browsers, to be assured everything was correctly implemented. The other browsers we tested were Mozilla Firefox 13.0, Opera 11.64 and Google Chrome 19.0.1084.56. The tests were performed under the same specifications pointed in the previous paragraph.

We also tested the application under an Android Ice Cream Sandwich Operative System (version 4.0.4), using the native Android browser.

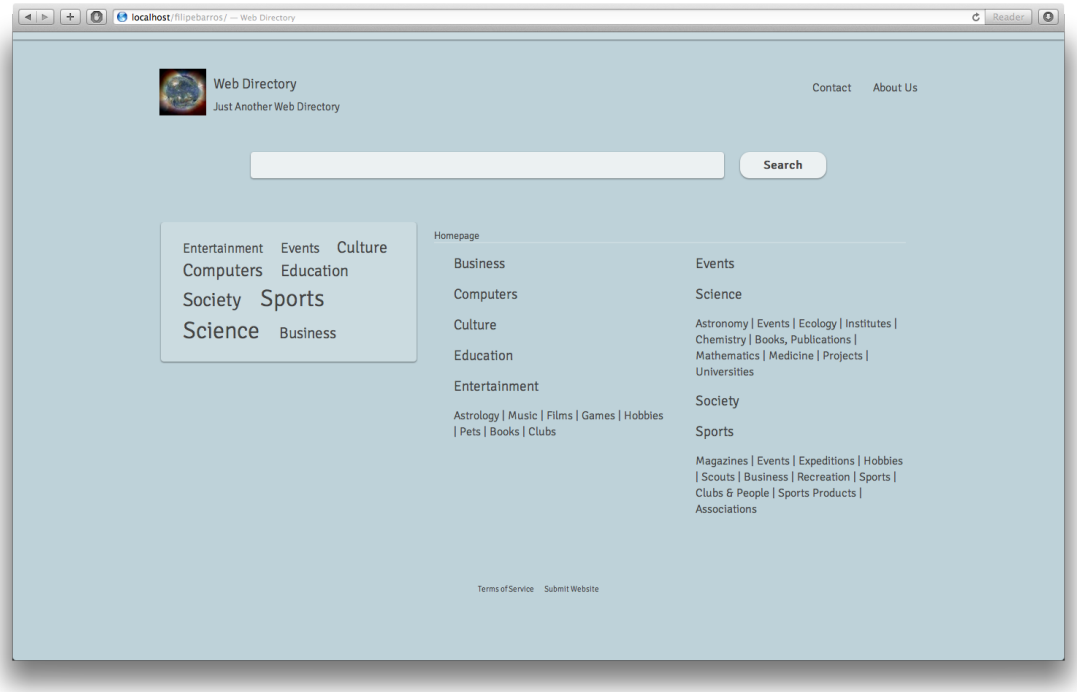


Figure 6.6: Web directory screenshot: interface of the homepage

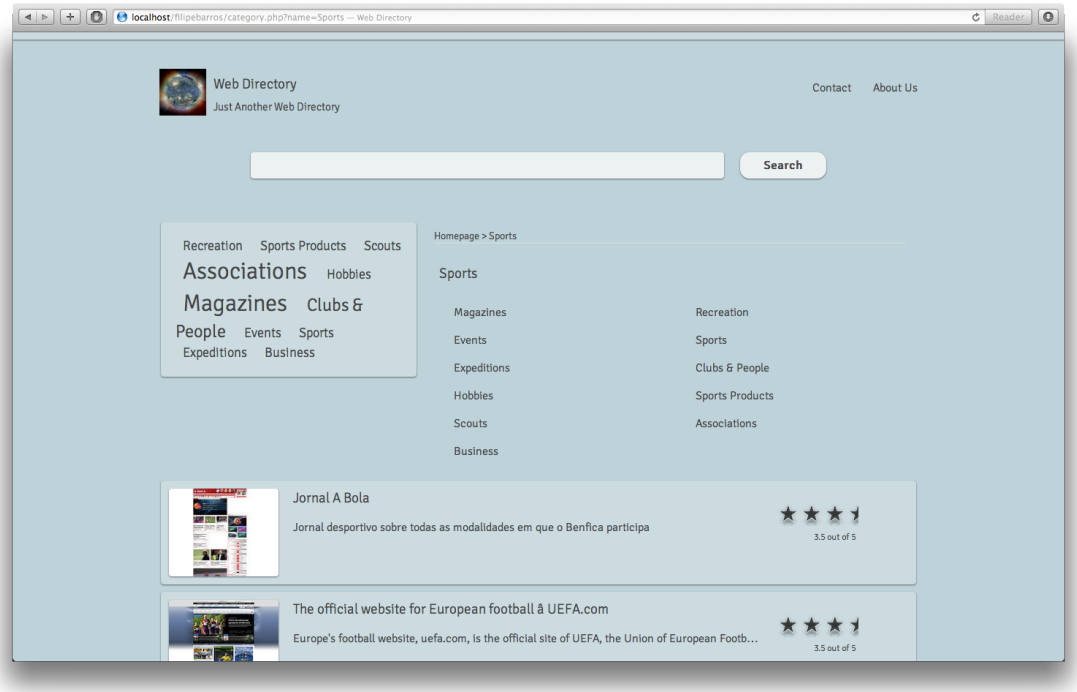


Figure 6.7: Web directory screenshot: the interface of a category page

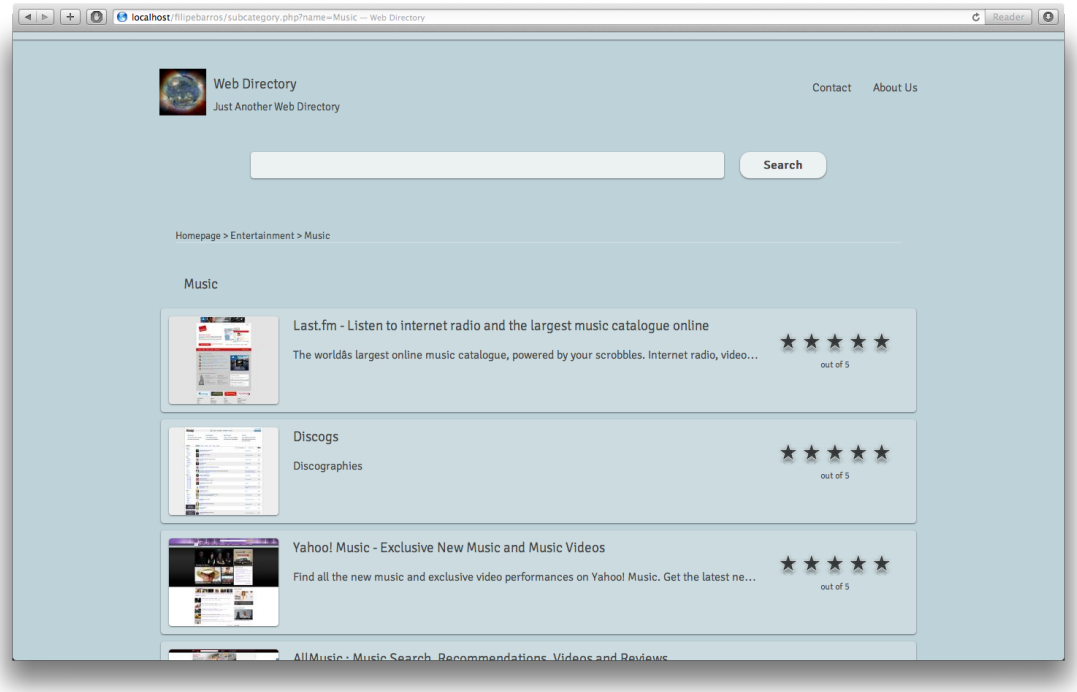


Figure 6.8: Web directory screenshot: the interface of a subcategory page

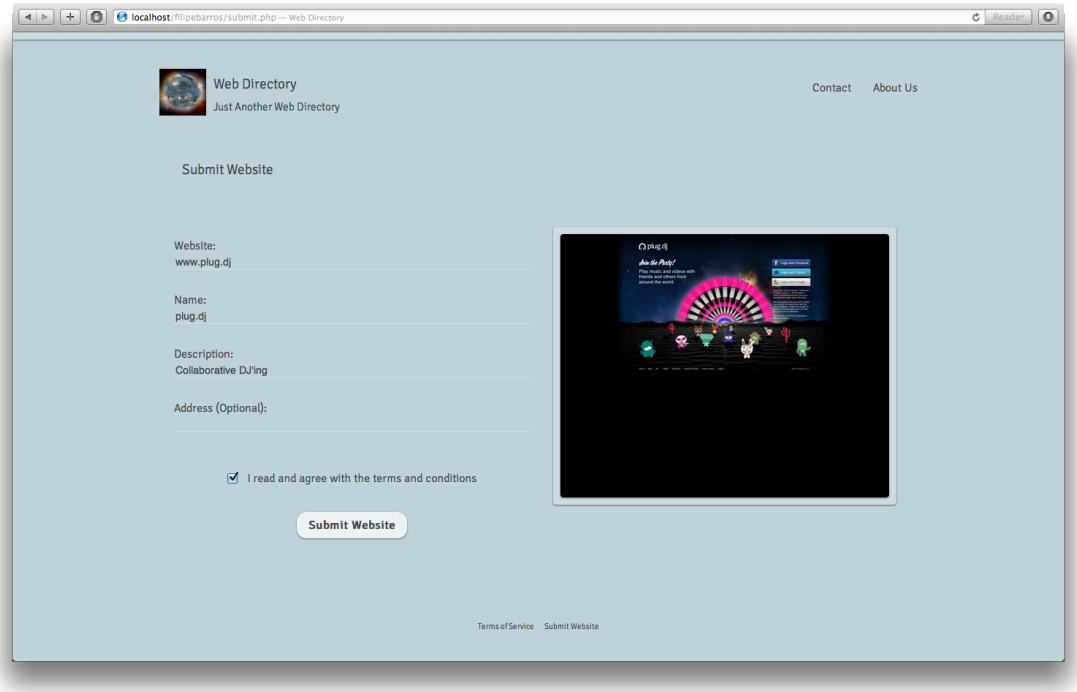


Figure 6.9: Web directory screenshot: the interface of the submit website page

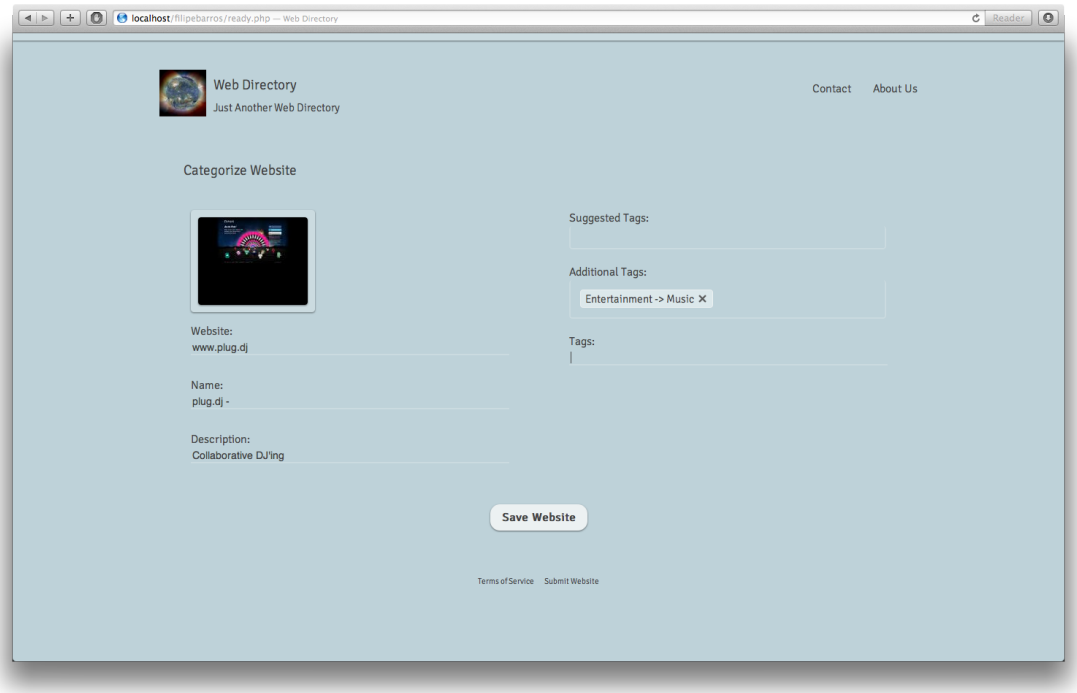


Figure 6.10: Web directory screenshot: the interface of the categorization page

Chapter 7

Results and Discussion

7.1 Usability Tests

In order to assert the scientific validity of this set of guidelines, we needed to elaborate a set of tests. To test the application, we had the help of seven people, asking them for feedback on the application and putting them executing some tasks. The task were to be executed without any help from us, but with clarifications whenever the user felt necessary. While the user was testing the application, the time was being measured by us.

In the end we asked what was the difficulty in a three level scale: easy, medium and hard.

7.1.1 Purpose of the Website

- What is your general impression of the website?
- What do you think you can do with the website?

7.1.2 Test Scenarios

1. Search for a website in the web directory (search engine or drill down);
2. Search for a website in the web directory (method not used in the first test);
3. Submit website to the web directory (add at least one Tag);

7.1.3 Conclusion

- What do you think about the web directory?
- What are the positive points? and the negatives?
- What you would like to have implemented as a feature?

In this part, our aim was to get to know more of what the user thought about the web directory, to address the negative points and get some ideas of the real users.

7.1.4 Prototype

- Would you like to have a feature to suggest you tags when adding a website to the web directory?
- Where would you like to see the map with the path to the website physical location?
- What do you think of this page? Would you like to have this to warn you when going to another page? Would you prefer to go directly to the page?
- What do you think of the breadcrumbs bar? Is it useful or not? Do you think it is a good idea to have it?

In this phase, we wanted to get the user inputs on some of the features that some users could have missed. When asking the questions we show the feature we are talking about (if implemented), so users could answer more accurately, and give their ideas about it.

The tests were made in a closed and controlled environment in order for the tester not to have any distractions. While the user was executing the actions, we were taking notes on the actions made by the user and what they were saying.

During the execution of the tasks, and whenever the user asked, we made some explanations of the task.

After the actions were executed, we asked the user about the things that he was saying and making, in order to understand their actions. During this phase, most of the users gave their input on the tasks, leaving us with just a few questions in the end about possible features and adjustments on the web application.

The script of the test can be found in appendix A.

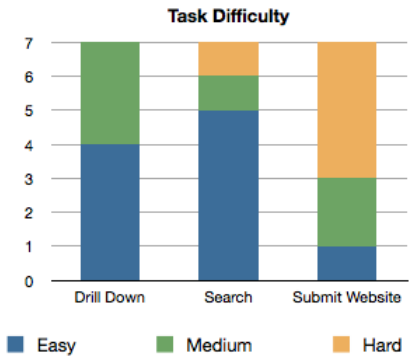


Figure 7.1: Results of the usability tests

7.2 Analysis

After the tests were done, we compiled them according with the responses we got (figure 7.1). We asked the reasons for the answers we got and we received a lot of answers, but a lot of them focused on the same points.

The test on the search, was only done to see the pattern of users that would choose the search engine first: five out of seven did. The difficulty of this test was related to some users that were using the drill down strategy and combined the search with it. It was giving all the results that matched the query, and they were trying to get the results that matched the query and were present in that category. Basically they were filtering the query, but it wasn't fully developed.

We had a lot of feedback from the users, all of them telling us the negative and positive aspects of the website. With this information we can provide a better solution. We are going to talk about the input received by the users, regarding each of the page where the users had to interact when testing.

7.2.1 Home Page

In the homepage (figure 6.6), A few users were complaining that the web directory looked too much like a company website, because of the logo and the navigation. They were saying the links in the upper part of the website had too much emphasis and should be smaller in order not to catch too much of their attention.

A bigger part said that the tag cloud should be bigger yet staying in the left part of the website. Their opinion is that they would firstly try to find the category they were searching in the tag cloud and only if they didn't find anything interesting there they would use the categories and subcategories.

The thing that most of the users complaint was because the link to submit a website was too small, and this proved to have a major impact in the difficulty of the third task: most of the users found it hard to find the button and this took them a longer time, compared to the action

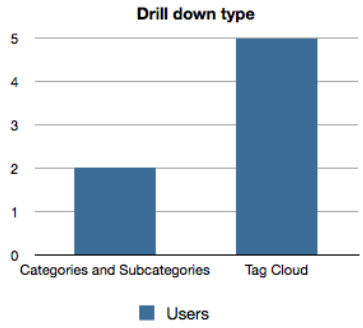


Figure 7.2: Type of drill down the user chooses: tag cloud or categories and subcategories list

of submitting. Most of the users approximately took one quarter of the time spent in the whole process just to find the the button. They demanded a bigger button and an advanced feature: when adding a website, they wanted to automatically add the tag of the category or subcategory they were browsing at that time.

7.2.2 Categories and Subcategories

When asked about the categories (figure 6.7) and subcategories (figure 6.8) pages, the users talked about the breadcrumbs. They had problems because of its position: they found it confusing because the tag cloud wasn't under it and they had doubts if the links of the tag cloud were related to the category they were viewing. Their solution would be to put the tag cloud below the breadcrumbs.

Regarding to the website links presentation and the redirection, there were two suggestions: to remove the address from the website area, because this takes too much space; and, when clicking the button, redirect to a page where the information of the website would be shown, including the address and a map showing how to go to the physical place of the website.

7.2.3 Submit Website Pages

The input we had regarding the submit pages (figure 6.9 and 6.10) was good. The users liked the automatic information extraction of the information when inputting a website, but they suggested the website not to allow any change while the data is fetched from the website and present a better loading message.

There was also a little obstacle in the second part of this action: in the tags division, some users were clicking in order to understand where to add a tag.

When asked about the automatic categorization, all users stated that it would be a good feature, but always giving the user the option to remove the tags the system could find related to the website. The reason is that the users could find that the tag was not accurate enough and could want to specify it.

7.2.4 Conclusions

The tests made were extremely useful in order to understand how the users use this kind of websites and how they think when browsing. These inputs were taken during the whole duration of the tests and not only in the end. With this we could ask the user the reasons why something was bad and have a better understanding of the problems they found.

7.3 Results Evaluation

From the results of the tests, we can infer that the design was well received by the users, but we can still improve the interface to make it even easier for the users to execute their tasks. Within this section, and to make it more understandable, we will talk about the improvements by tasks, instead of pages.

7.3.1 Drill Down Through Categories and Subcategories

When drilling down through the categories and subcategories, the biggest problem was due to the breadcrumbs positioning that was causing confusion in a few users. According to the users, a solution would be to put the tag cloud under the breadcrumbs, promoting the separation of content, thus becoming more clear that the tag cloud was related to the category.

7.3.2 Search Engine

About this we can only say a few things, since this is not fully implemented. The input we had was about filtering automatically the search, depending on the category we were browsing. This doesn't mean discard information unrelated to the category, but reorganize the information giving a bigger weight to the information related to the category from which the search was done.

7.3.3 Submit Website

This is the part that the users said that needed more changes due to the confusion and misleads it gives.

The button to submit the website should be bigger, because most of the users had problems finding it, even scrolling down and up, trying to find something. After some seconds, and without any help, they managed to find it but it is still a flaw that needs to be corrected.

Another problem that needs to be addressed is the fact the website allows interaction when loading information about the website that is being submitted. In order not to allow the interaction with the website, we can use JavaScript to block the interactions.

A few users also mentioned the use of an alternate message, more explicit than the current one.

7.3.4 Task Transversal

The problems related to the execution of one task were listed, but there is a problem that is related to every task: the website button. As the website button leads us to another page, some users said that, instead of showing one page with almost no content, warning the user he was leaving the website was annoying, they said that it would be better to show some detailed information of the website there, instead of showing too much condensed information on the button.

With this, we could be showing the user more information, while still warning the user of the fact they were leaving the website. With this, we could manage to show all the information about the website on this new redirect page, and minimize the amount of text on the website button.

Chapter 8

Conclusions

8.1 Results

With the tests we made, we managed to get a few guidelines related to web directories. These guidelines don't have the objective to replace the guidelines that can be found in the website *www.usability.gov* [oHSA06], this is just a complementary set of guidelines specific to the development of web directories.

Along with the guidelines that are defined in [oHSA06], we have some more that are more defined for web directories and its structure. These guidelines we propose were extracted from the tests we done with the users.

- Provide a search engine to the user! A great part of the users, during the usability tests, chose to use the search engine first, instead of drilling down through the tag cloud or the categories and subcategories list.
- Get the emphasis on the tag cloud! The users preferred to drill down using the tag cloud, instead of the categories and subcategories.
- Do not give navigation links more emphasis than they deserve. It shouldn't be too catchy, or the user will lose sight of what really matters on the site.
- Clearly define the boundaries between different sections. If the sections are not clearly defined, the users will be confused.
- Do not overload the sections with information. It's better to provide the information after the user clicks the link. Sometimes is better to have an extra click and show information better.
- Automatic categorization is a very important feature. All the users would like to see this feature implemented, since it would make it easier for the users to categorize the website.

- Whenever necessary and possible, use already built JavaScript libraries and/or plug-ins. This spares us the time to implement them, and normally they are better developed and maintained, because of the huge community that develops and maintains these libraries plug-ins.

8.2 Future Work

Outside of the scope of the project, there were two features that were researched in order to possibly add to the web directory, but due to the time spent in the development and writing the thesis, these features weren't added. Despite this, we tested one of this features and even asked the users about the other one. The search engine is a must in every web directory due to the nature of the users nowadays and also because this is a very good way to spare the time searching. The automatic categorization of the websites added to the web directory was not tested, but, when asked about the possibility of integration with this feature, all the users said that this would be a great feature.

In the next sections will be given a brief contextualization and explanation of the feature and the inputs we got from the users, when testing the web directory.

8.2.1 Automatic Categorization

In web directories like Yahoo! Web Directory (figure 3.1) or the Open Directory Project (figure 3.2) where there is a large team supporting and contributing, the whole process of addition is done manually, including the categorization of the website, but, with the number of websites that exist nowadays, even with a large team supporting, there are some processes that can be done automatically, like the categorization of the website.

Based on the `<meta name="keywords">` we can sort the categories and/or subcategories of a website. This is only intended to be a suggestion and not an automatic categorization without the user input.

There are some requirements for this to work: the website needs to have the `<meta name="keywords">` defined, for us to use this. If the website doesn't have this, then we can't generate the suggested categorizations, but the user can categorize the website manually.

We also have to deal with some websites where they put website unrelated keywords to appear in more searches.

The suggested categorizations could be removed if the user thinks that they are not correct or accurate. The user can also add other tags manually.

Of course that, having the user input, the objective is for the algorithm to learn from the users input in order to deliver better results in the long term.

8.2.2 Search Engine

Web directories were once the main way to search for websites on the internet, but due to its expansion of the web, it's nearly impossible to have all the websites categorized in one place and, even if possible, it would be extremely hard for the users to find anything.

A different approach was made, using a search engine, where a user only needed to use some keywords to find websites related to the keywords inputted.

Since nowadays most of the users are "search oriented", this is a feature that most of the users want implemented. During this phase there wasn't enough research to suggest a good way to do it.

From the input we got we can say that users would like to search depending on the level they are browsing: if they are in the homepage, they want all the results and if they are browsing the categories, they want to, at least have the subcategories related to that categories first.

Appendix A

Usability Tests Script

In order to do the tests we had to prepare a script to guide us during the whole phase. This script was followed with the seven users that were helping us.

Some of the users used an adaptation on the script: while drilling down through the categories, they used the search bar. While this wasn't in our previsions, this gave us ideas on adaptations to make in the web application.

One of the adaptations was to use the search engine in the scope the user was in. For example, if the user was in the home page, show the categories first, and the subcategories after, but if the user was in the categories, show the subcategories related to the categories first, and only after show the other results.

Usability Tests Script

1. Introduction

1.1. Purpose

You have been invited for this session to help us do some tests on our website, which is still in development. Your help will be very useful to improve and validate our ideas. We are here testing the tool, not you, you can't do anything wrong here. First we would like to know some things about you.

1.2. User Profiling

- Age
- Occupation
- Education
- Experience using web directories.

We are going to ask you some questions and do the tests. We would like to ask you to explain your thought process aloud. We are going to be taking notes of your interactions with the website and your thoughts. As we told you, there's nothing here you can do wrong, and there are no right or wrong questions and your feedback will be very valuable for us.

2. Purpose of the Tool

- What do you think is the purpose of this website?
- What do you think you can do in this website?

3. Tests

We ask the user about one of his interests firstly, and then ask him, looking at the website, what would he do to find a website related and waiting for his input.

Depending on the way he would do, we would make a different test: drill down through the categories and subcategories or use the search bar.

Tell the user the scale: easy, medium and hard.

3.1. Drill Down

- Ask the user to drill down until he reaches one website given to him.
- Watch the if the user uses the tag cloud or the categories and subcategories.
- When the user ends, ask for his input and ask for the difficulty.

3.2. Search

- Ask the user to write the query he thinks that is better to find the website.
- When the user ends, ask for the difficulty of using this.

3.3. Submit Website

- Find the link to the submit webpage.
- Write the website on the input box.
- Submit website.
- Select tags.

4. Conclusion

- What do you think about the website?
- In a scale of easy, medium and hard, how you you classify the website?
- What do you think would be a good feature?

5. User Ideas

- What do you think of having a mechanism to automatically categorize the websites?
- What do you think of having a redirect page? Is it a good idea?
- Do you think that the use of breadcrumbs is good? Is it intuitive and easy to use?
- How would you show the map of the path to the physical location of the website?

Figure A.1: Script used in the usability tests

References

[Awi12] Awio Web Services LLC. Web Browser Market Share Trends. http://www.w3counter.com/trends, May 2012. last access: Jun 10, 2012.

[Ber89] Berners-Lee, Tim. Information Management: A Proposal. http://www.w3.org/History/1989/proposal.html, March 1989. last access: Jun 02, 2012.

[Ber92] Berners-Lee, Tim. HTML Tags. http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html, November 1992. last access: Jun 01, 2012.

[Ber93] Berners-Lee, Tim and Connolly, Daniel. Hypertext Markup Language(HTML) — A Representation of Textual Information and MetaInformation for Retrieval and Interchange. http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt, June 1993. last access: May 27, 2012.

[Ber95] Berners-Lee, Tim and Connolly, Daniel. Hypertext Markup Language - 2.0. http://www.w3.org/MarkUp/html-spec/html-spec_toc.html, September 1995. last access: Jun 05, 2012.

[Bre08] Brewer, Dustin. Why web standards are important in web design. http://dustinbrewer.com/why-web-standards-are-important-in-web-design/, January 2008. last access: Jun 08, 2012.

[Dee10] Deep Blue Sky. HTML5 CSS3 Support. http://www.findmebyip.com/litmus/, July 2010. last access: Jun 10, 2012.

[Fri07] Friedman, Vitaly. Tag Clouds Gallery: Examples And Good Practices. http://www.smashingmagazine.com/2007/11/07/tag-clouds-gallery-examples-and-good-practices/, November 2007. last access: Jun 14, 2012.

[Fri10] Friedman, Vitaly. Web Design Trends 2010: Real-Life Metaphors and CSS3 Adaptation. http://www.smashingmagazine.com/2010/05/20/web-design-trends-2010-real-life-metaphors-and-css3-adaptation/, March 2010. last access: Feb 01, 2012.

[Goo] Google. Google Maps API — Google Developers. https://developers.google.com/maps/. last access: Jun 11, 2012.

[Ham] Hammond, Paul. webkit2png. http://www.paulhammond.org/webkit2png/. last access: Jun 10, 2012.

REFERENCES

[Hun] Hunt, Ben. Current Style in Modern Web Design. http://www.webdesignfromscratch.com/web-design/current-style/. last access: Dec 02, 2011.

[Jon10] Jones, Brandon. The State of Web Design Trends: 2011 Annual Edition. http://webdesign.tutsplus.com/articles/design-theory/the-state-of-web-design-trends-2011-annual-edition/, December 2010. last access: Dec 02, 2011.

[Kru05] Steve Krug. *Don't Make Me Think: A Common Sense Approach to the Web (2nd Edition)*. New Riders Publishing, Thousand Oaks, CA, USA, 2005.

[LB99] Hakon Wium Lie and Bert Bos. *Cascading Style Sheets: Designing for the Web*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1999.

[Net12] Netscape. ODP - Open Directory Project. http://www.dmoz.org, 2012. last access: Jun 15, 2012.

[Nie93] J. Nielsen. Iterative user-interface design. *Computer*, 26(11):32 –41, nov. 1993.

[Nie99] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Thousand Oaks, CA, USA, 1999.

[Nie00] Nielsen, Jakob. Why You Only Need to Test with 5 Users. http://www.useit.com/alertbox/20000319.html, March 2000. last access: May 21, 2012.

[Nie05] Nielsen, Jak. Ten Usability Heuristics. http://www.useit.com/papers/heuristic/heuristic_list.html, 2005. last access: Mar 2, 2012.

[NuS] NuSphere. PHP History. http://www.nusphere.com/php/php_history.htm. last access: Jun 10, 2012.

[oHSA06] United States. Dept. of Health, Human Services, and United States. General Services Administration. *Research-Based Web Design & Usability Guidelines*. U.S. Department of Health and Human Services, 2006.

[RLAK98] Dave Raggett, Jenny Lam, Ian Alexander, and Michael Kmiec. *Raggett on HTML 4 (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[Tea10] Teague, Jason. The Evolution of HTML & CSS. http://www.jasonspeaking.com/index.php/2010/04/the-evolution-of-html-css/, April 2010. last access: Jan 29, 2012.

[The12] The jQuery Foundation. jQuery: The Write Less, Do More, JavaScript Library. http://jquery.com, 2012. last access: Jun 11, 2012.

[Vee00] Jeffrey Veen. *The Art and Science of Web Design*. Pearson Education, 2000.

[W3Ca] W3C. About W3C. http://www.w3.org/Consortium/. last access: Jun 09, 2012.

[W3Cb] W3C. Browsers and Authoring Tools - W3C. http://www.w3.org/standards/agents/. last access: Jun 09, 2012.

[W3Cc] W3C. Semantic Web - W3C. http://www.w3.org/standards/semanticweb/. last access: Jun 09, 2012.

[W3Cd] W3C. Standards - W3C. http://www.w3.org/standards/. last access: Jun 09, 2012.

[W3Ce] W3C. W3C Current Members. http://www.w3.org/Consortium/Member/List. last access: Jun 10, 2012.

[W3Cf] W3C. Web Architecture - W3C. http://www.w3.org/standards/webarch/. last access: Jun 09, 2012.

[W3Cg] W3C. Web Design and Applications - W3C. http://www.w3.org/standards/webdesign/. last access: Jun 09, 2012.

[W3Ch] W3C. Web of Devices - W3C. http://www.w3.org/standards/webofdevices/. last access: Jun 09, 2012.

[W3Ci] W3C. Web of Services - W3C. http://www.w3.org/standards/webofservices/. last access: Jun 09, 2012.

[W3Cj] W3C. XML Technology - W3C. http://www.w3.org/standards/xml/. last access: Jun 09, 2012.

[W3C11a] W3C. CSS Color Module Level 3. http://www.w3.org/TR/css3-color/, June 2011. last access: Jun 04.

[W3C11b] W3C. CSS Namespaces Module. http://www.w3.org/TR/css3-namespace/, September 2011. last access: Jun 04.

[W3C11c] W3C. Selectors Level 3. http://www.w3.org/TR/css3-selectors/, September 2011. last access: Jun 04.

[WHA11] WHATWG. HTML5: A technical specification for Web developers. http://www.whatwg.org/specs/web-apps/current-work/multipage/introduction.html#history-1, 2011. last access: Jun 03, 2012.

[Wik12] Wikipedia. Dot-com bubble. http://en.wikipedia.org/wiki/Dot-com_bubble, June 2012. last access: Jun 15, 2012.

[Wil] Wilton-Jones, Mark. JavaScript history. http://www.howtocreate.co.uk/jshistory.html. last access: Jun 07, 2012.

[Wis] Wise, Cheryl. History of HTML CSS. http://www.wdtonline.com/wdtMagazine/MemberWorks/WiserWays/csshtml.htm. last access: Feb 01, 2012.

[Wiu94] Wium Lie, Håkon. Cascading HTML style sheets — a proposal. http://www.w3.org/People/howcome/p/cascade.html, October 1994. last access: Jun 07, 2012.

[wor] WorldWideWeb (The First ever Web Browser). http://www.googlechromedownload.com/worldwideweb-the-first-ever-web-browser/. last access: May 20, 2012.

[Yah12] Yahoo! Inc. Yahoo! Directory. http://dir.yahoo.com, 2012. last access: Jun 15, 2012.

[You10] Young, Alex. History of JavaScript: Part 1. http://dailyjs.com/2010/05/24/history-of-javascript-1/, May 2010. last access: Jun 10, 2012.