BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

COMPUTER SCIENCE, ENGLISH

DIPLOMA THESIS

Agile Web Development with RubyOnRails and Web Services

Supervisor:

Lecturer Phd. COJOCAR Grigoreta

Author:

POP Bogdan

2010

Abstract

This paper presents a new concept system that allows its users, either friends or strangers, to interact with each other, share what restaurants, cafes or pubs they've visited, how the service was or how much they've paid. The system will combine a Web application with an iPhone native application which can both be used to find a location suited for users' desires. They will be able to search for restaurants around themselves, by name or address, while filtering the results based on ratings, food and beverage pricing.

The paper is structured as follows. Chapter 1 presents the history, evolution and development of the Internet and the World Wide Web, to clearly separate the two different concepts. It also presents the characteristics of smartphones and multi-touch devices. Chapter 2 presents different concepts and technologies that can be used to develop Web and mobile applications, as well as novel technologies that can be used to build great Web and mobile applications. Chapter 3 presents RubyOnRails, a powerful Web development framework which was used to build numerous successful Web applications. Agile software development is briefly described as RubyOnRails incorporates in its features many agile practices. Chapter 4 presents comparisons between different similar technologies while showcasing the advantages and disadvantages each of them have. Chapter 5 describes how a RubyOnRails Web application can be extended by using a native iPhone application while the 6th and final chapter presents the actual Web / mobile system, from concept to key programmable features.

Table of contents

1. The Internet, the Web and Smartphone Devices

The two terms: the *Internet* and the *World Wide Web* are used on a daily basis without much distinction. However, the Internet and the World Wide Web are two completely different systems.

Briefly described, the Internet is a global system of computers interconnected within small or large computer networks connected between each other by using the TCP/IP protocol. It serves billions of people worldwide. Moreover, the Internet serves as a hardware and software infrastructure which is used by one of the services which runs on top of it: the World Wide Web (WWW or Web).

The Web is a collection of interconnected documents and other resources, linked by hyperlinks and URLs (Uniform Resource Locator). These links are the core of the Web, because removing them would result in a useless service, having billions of inaccessible, hidden resources around worldwide Web servers. The link is a simple concept, and it is one of the primary forces driving the success of the Web. A link has two ends: a source and a destination with a direction pointing from source to destination [1].

Merging computers and communications has changed the way computer systems are organized. The concept of the "computer center" as a room with a powerful computer where users go to have their documents processed is obsolete. The old model of a single computer serving one organization's computational needs has been replaced by a model in which a number of separated but interconnected computers do the job. These systems are called computer networks. The Internet is not a single computer network, but a network of networks, while the Web is a distributed system that runs on top of the Internet. The difference between a computer network and a distributed system, such as the Web, is that the distributed system appears to its users as a single coherent system [2].

For example, to non-technical users, everything in the World Wide Web looks like a document or a Web page. Non-technical users don't know that these documents are spread across the globe, hosted on Web servers and that when they click a link in a Web page their request is sent and passed on through tens or hundreds of computers over the Internet.

A smartphone is a device that runs its own operating system which usually has a powerful processor capable of running complex applications in a manner similar to how a standard PC would. Smartphones usually combine a phone that has Internet communications features such as email and browsing, media players and more.

The first smartphone was introduced in 1992, one year before the first Web site was launched at CERN (European Organization for Nuclear Research), and the development of both has seen the same growth since then.

1.1 The Dawn of the Internet and the World Wide Web

Today’s Internet was born due to the United States Air Force need to maintain its technological lead. The US Government created the Advanced Research Projects Agency (ARPA) in February 1958, which in turn created Information Processing Technology Office (IPTO).

Roughly a decade later, collaborations between IPTO, MIT (Massachusetts Institute of Technology) Lincoln Laboratory and UCLA (University of California, Los Angeles) have resulted in a set of theoretical foundations for packet networks and for hierarchical routing, concepts which have been the foundation of today's Internet.

In 1968 Lawrence Roberts published a report called Resource Sharing Computer Networks which laid the foundation for the launch of the working (Advanced Research Projects Agency Network) ARPANET the following year. The report described a high capacity, reliable communication system with multiple points and fast response, which were required for an interactive computer network. The protocol described was the result of discussion taken with interested parties in the previous year [3].

The ARPANET was the world's first operational packet switching network and is the foundation of today’s Internet. The connection between the first two points was established in October 1969 and by the end of 1971 there were around 15 different sites interconnected using the ARPANET, including many US university research labs. By 1981, networks have expanded towards Canada, Hong Kong or Australia [4].

Two years later, on the 1st of January 1983 all hosts on the ARPANET were switched from the older NCP (Network Control Program) protocols to the newly developed TCP/IP (Transmission Control Protocol / Internet Protocol) protocol which was specifically designed to assure reliable end-to-end byte streams over unreliable Internetworks. An Internetwork differs from regular, single networks because different hardware parts may have different topologies, delays, bandwidths, packet sizes and other parameters. TCP protocol was designed to dynamically adapt to different Internetworks and to be robust while facing different kinds of failures [2].

In 1985 the United States National Science Foundation (NSF) ordered the development of NSFNET, a 56 kilobit/second network backbone that would connect major universities. By the year 1989, NSFNET was interconnected with MCI Mail, the first commercial system connected to NSFNET. Soon after, other major commercial services were connected, and first Internet services providers were incorporated, such as: PSINet, UUNET, with PSINet being the first to provide commercial Internet access to companies.

Although highly used in research labs, universities and government institutions, the network didn't gain a public face until the 90s. This was due to the fact that Sir Timothy John "Tim" Berners-Lee invented the World Wide Web (Web) in 1989.

The first name of the Web when Tim Berners-Lee was working on it was Enquire, short for "Enquire Within upon Everything" – an old Victorian book. Enquire, the first Web like program was developed by Tim Berners-Lee in his spare time and for personal usage, after different thoughts and ideas, disparate conversations and disconnected experiments conducted at CERN. Enquire was developed to help its creator remember connections among various people, computers and projects at the CERN labs. This vision has taken firm roots in his consciousness.

"Suppose all information stored on computers everywhere were linked. Suppose I could program my computer to create a space in which anything could be linked to anything! [] The vision I have for the Web is about anything being potentially connected with anything!" – Tim Berners-Lee [5].

With such a system all the information in every computer at CERN and on the planet would be available to anyone, everywhere they're located. This way there would be a single, global information space. Once any bit of information would be labeled with an address, one could tell his/hers computer to get it. By being able to reference anything with ease, a computer could represent associations between things and make them share a relationship. A Web of information would form [5]. This was the goal of the Web, and current development and concepts embraced by developers worldwide as Web 3.0 is a huge step towards that goal.

In March 1989, Tim Berners-Lee published a paper "Information Management: A Proposal" that concerned the management of general information about accelerators and experiments at CERN. Tim Berners-Lee's breakthrough was to combine the hypertext to the Internet. He developed a system of globally unique identifiers for resources on the Web. He developed the Universal Document Identifier (UDI), today's Uniform Resource Locator (URL) and Uniform Resource Identifier (URI). He also drafter and developed the publishing language HyperText Markup Language (HTML) and the Hypertext Transfer Protocol (HTTP) [6].

The Web required only unidirectional links rather than bidirectional ones. This made it possible for someone to link to another resource without action by the owner of that resource.

On April 30, 1993, roughly one year after first Web site available at CERN [7], they announced that the World Wide Web would be free to anyone, with no fees due [8]. One year later, Tim Berners-Lee left CERN and founded World Wide Web Consortium (W3C). The organization's Website is the first ever to be available since the 6th of July 1994.

By the end of 1994, although the total number of Web sites was still low compared to present standards, quite a number of notable Web sites were already active, many of which are the precursors or inspiration for today's most popular services.

The Web enabled the spread of information over the Internet through a flexible and easy-to-use medium. Therefore, it played an important role in popularizing use of the Internet. Sir Timothy John "Tim" Berners-Lee played a key role in guiding the development of Web standards. In recent years he has advocated his vision of a semantic Web.

The first Web browser was W3C “Line Mode” browser [9], which had a command line interface, while the first Web browser with a graphical interface was called “WorldWideWeb” (Figure 1). The first popular Web browser was Mosaic, which was also the foundation of the Netscape browser.

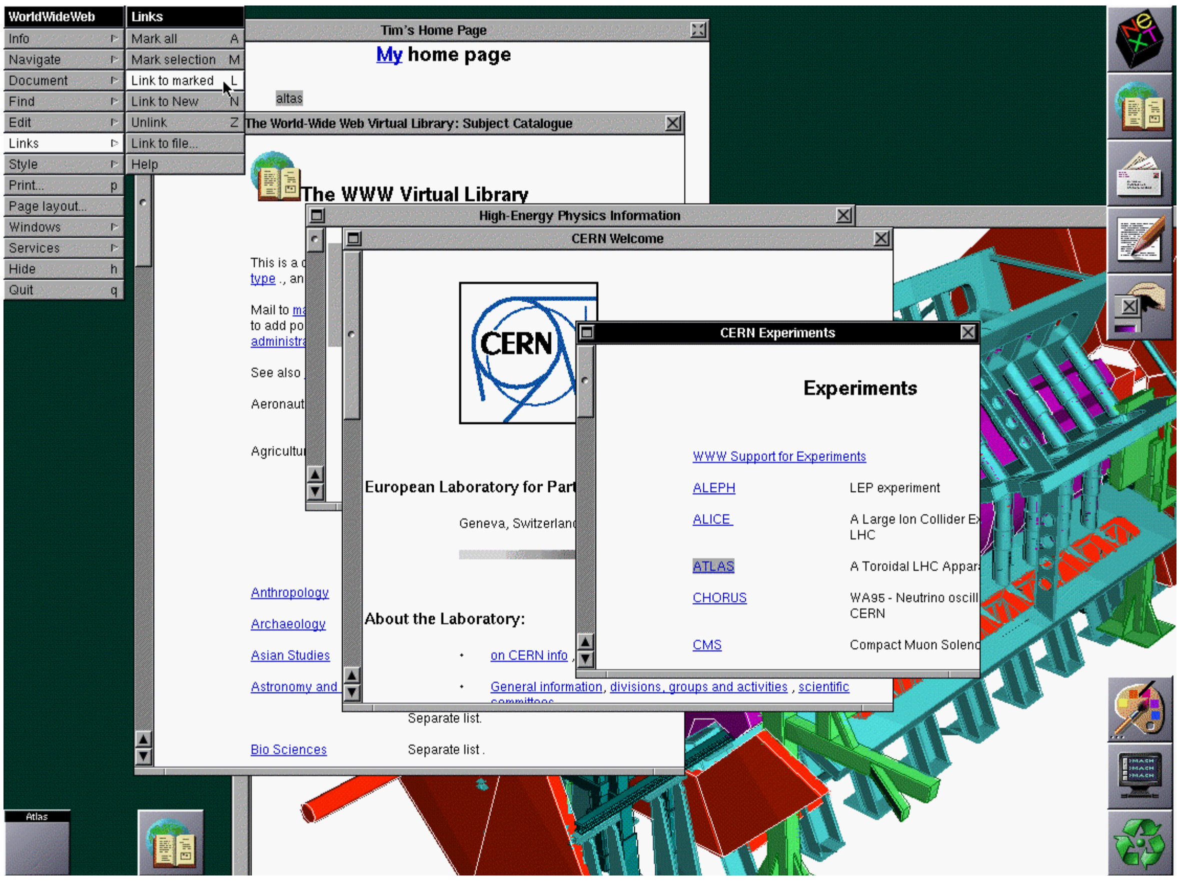


Figure 1. WorldWideWeb Web browser, picture from www.w3.org

1.2 Smartphone Devices

The world’s first smartphone, Simon, was developed and showcased to general public by IBM, in 1992 [10]. Since then, various companies have presented different concepts for smartphones, but one concept that has established in the last couple of years is the multi-touch device, with the first of this kind being Apple’s iPhone first version, released in 2007 (Figure 2).

“This is a day I’ve been looking for two and a half years. Every once in a while a revolutionary product comes along that changes everything. [] Today we’re introducing three revolutionary products. The first is a widescreen iPod with touch controls. The second is a revolutionary mobile phone. And the third is a breakthrough Internet communications device. These are not three separate devices. This is one device, and we are calling it iPhone. Today Apple is going to reinvent the phone!” – Steve Jobs [11]

From that point onwards, smartphones that had QUERTY keyboards built into their hardware started to lose market share and leave great opportunities for multi-touch devices. For example, in the 1st quarter of 2007, all smartphones had a QUERTY keyboard and complex, hard to use interfaces [12]. One year later, Apple’s iPhone had 5% of the smartphone market share. The release of HTC touch, Nokia’s touch devices and others has triggered a rapid growth of touch devices, which reached 36.2% of the smartphone market share in the 1st quarter or 2009 [13].

The difference between a smartphone device and a regular cell phone is that the first runs on a native operating system, resulting in tools, features and abilities developers can exploit to create native, complex applications. On the other hand a regular mobile phone is only capable to run simple applications based on generic platforms such as Java ME (Java Platform, Micro Edition) or BREW (Binary Runtime Environment for Wireless) [14].

Figure 2. 4th iPhone version released June 2010 (www.apple.com)

An important difference between a keyboard smartphone and a touch device is the fact that the interface of the second can be updated and improved with minor changes to its operating system and software, while the first category of smartphones require hardware changes to be paired with software changes. One cannot add a new feature that uses a new button without having a new button on the phone. On a touch device, adding new buttons is only a software dependant task.

Based on its producer, each touch device has different operating system, hardware features, or APIs. Hence, there are a lot of operating systems which support various programming languages one can use to develop mobile applications.

Today's most popular and known mobile development platforms are Java ME, Symbian, Android, .NET Compact Framework for Windows CE (Windows Embedded Compact), Windows Mobile, Palm OS and the iPhone OS (iOS since June 7, 2010).

2. Future of Web and Mobile Applications

There has never been a better time to be in the mobile and Web industry. Development frameworks have been around for a couple of years now. However, since the launch of mobile devices such as the iPhone, the iPad or Android, simply developing for the Web doesn't mean that all users will be satisfied. People travel a lot and when they do so, they need to use different Web applications on mobile devices, and a native application is always better than a Web application viewed on a mobile browser.

HTML5 introduced many technology advancements such as Client Side Storage (Web Storage), Web Sockets communication that support full-duplex channels just as mobile phones and Desktop Experience through Notifications, Drag & Drop and Geolocation, all of which are standards proposed by World Wide Web Consortium [15].

Moreover, the days of building just for the Web are almost over. With the imminent release of Google's ChromeOS and the wider adoption on WebOS, developers need to create applications that will work on these platforms [15].

Finally, a shift in data storage has been triggered by successful companies such as Twitter, Digg or Facebook have brought NoSQL (Not Only SQL) databases to the spotlight. NoSQL databases define class of non-relational data stores which may not require fixed table schemas and usually avoid join operations and typically scale horizontally [15].

2.1 The Dot-Com Bubble, Web 2.0 and Web 3.0

The dot-com bubble was a 5 year period which started in 1995 during which stock markets have seen their indices rapidly raise, most of them being triggered by the developing Internet sector which had a great commercial growth. During this period a lot of Internet based companies were founded, all of which were referred to as dot-coms. Other regular companies saw their stock prices go up only if they added an *e-* prefix to their name and a *.com* top level domain (TLD) [16].

Therefore, during this period, huge numbers of domains were registered. This resulted in a huge need for Web development services and new companies, services and jobs were created.

The dot-com bubble period was one when every company owner wanted their name to be on the Internet, so a large number of simple, basic, brochure type Web sites were developed. Apart from these simple Web sites, which had static content and their pages written in HTML and were styled with basic CSS (Cascading Style Sheet) and contained few images, a couple of dynamic Web sites, or Web portals, were launched.

Since then a lot of Web development technologies have been released and gone obsolete. Since then Web site development has evolved and today a typical Web site has dynamic content, which its owners can update online using a Web text editor. Other Web sites function as forums, chat rooms, search engines, online stores, blogs and in many other ways.

In August 2003, MySpace was launched. One year later came Facebook and Flickr. In 2005 YouTube was launched and 2006 was the year when Twitter became available. All these Web sites are different than the rest. They don’t offer information which is edited by a single company or individual. They offer content that’s being contributed by their users, their communities. The content users can create has no limit. It can range from short bursts of text (known as microblogging - Twitter), to images (Flickr), videos (YouTube), media (MySpace) and more.

These Web sites are in fact Web applications that ease information sharing and collaboration through interactive user-centered designs. Such a Web application has been defined as being Web 2.0 since Tim O'Reilly referred to it 2005 [17].

Web 2.0 Web sites can be grouped in different categories based on their primary goal or content type: hosted services, Web applications, blogs, wikis, social-networking sites, video-sharing sites and more [18].

But the Web isn’t stopping. The Web 2.0 concept isn’t established, yet another concept, Web 3.0 is picking momentum in the Web development communities worldwide. However, the Web 3.0 definitions are sparse and some of them appear as science fiction stories today.

Some say that Web 3.0 is about a semantic Web and personalization [19], some say that the semantic Web is not achievable and Web 3.0 is doomed before taken into serious consideration [20]. People having a business background say that Web 3.0 is nothing but Web 2.0 Web sites with a profitable business model [21]. The strangest, futuristic concepts are that Web 3.0 will be a Web where computers will be generating new information instead of human beings, and that Wolfram Alpha is the search engine of the future [22]. Other concepts describe a Web where users type sentences, not URL into their browsers and the Web does the rest: searching, preparing information, displaying it. In this scenario the Web will act like a personal assistant. Another concept would be a Web that combines current Web 2.0 concepts with 3D space, augmented reality, Geolocation and other technologies [23].

Tim Berners-Lee, inventor of the Web, is advocating towards a Web 3.0 as a semantic Web. In a recent conference he presented how information published on the Web in raw, interchange format can generate huge improvements in people's everyday lives. For example, United Kingdom's government has published raw data on bicycle accidents. Two days later a Web application by Times Online was showing real-time mashups about bicycle accidents in the UK. GeoEye has released satellite imagery days after the Haiti disaster in January 2010 which were used by the open source community to create maps and mashups. By the time rescue teams arrived at the location, they already knew precise locations of refugee camps, destroyed buildings, blocked roads and more. And this is just the beginning [24].

2.2 Technologies and Concepts Used in Web Development

Web development is similar to desktop software development, but each of them has its own characteristics. For example, desktop software usually requires installation before usage, installation which could take from seconds to entire hours to perform. On the other hand, Web applications and Web sites do not require installation from the user. The usual roadblock users could hit while using a Web application or site is a register / login procedure.

One important issue with Web development is memory footprint of Web applications and Web sites. While desktop applications can have large memory footprints, as users know what they get into when they buy a specific software product, Web application are completely different, as most of the processing is done server side.

Therefore, memory required by the Web application has to be kept at a minimum. This is because a Web application may run in a shared environment, on a virtual server, or on a dedicated server which has its own limits. Developers cannot predict how popular a Web site or Web application can become, how many people will use it simultaneously and how peak usage hours will look like [25].

Having a large memory footprint on a Web application that runs on a shared environment, classic or virtual server, may result in account suspension triggered by the Web host, because the application is having a detrimental effect on other Web sites and Web application the provider hosts. If such a scenario should happen, the implications are huge. Data and financial loss are only two of them.

Moreover, having a large memory footprint on a Web application that runs on a dedicated server could result in halting the server during peak hours, unresponsiveness of the application during high loads on the server, with direct impact on user experience.

Finally, having a large memory footprint usually affects the costs of hosting the application. Larger footprint results in larger amounts of required RAM and burstable memory, resulting in higher costs.

While desktop applications must be compatible with different platforms and operating systems, Web applications and Web sites must be compatible with different platforms, operating systems, Web browsers and different versions of the same Web browser.

Therefore, a Web application or Web site should comply with all current standards and also handle custom requirements by different browsers, such as quirks mode for Internet Explorer which doesn’t fully support W3C and IETF (Internet Engineering Task Force) standards [26].

A Web application should also be responsive and fast to load. One that has a slow interface and page loading times spanning multiple seconds will likely fail. This concepts extends to

Web forms used in Web development, which should only ask for essential information and have as few fields as possible [27,28].

The design and user interface of a Web application is also important, as it can be the deciding factor in application usage or dropout. For instance, layouts affect usage. Browsers can be resized, viewed full screen, on small resolutions and so on. Choosing between fixed, fluid, hybrid, adaptive or elastic layouts should be done carefully [29].

Typography and color usage also plays an important aspect in Web application and site design. Typography and color can be used to point users in the right direction, towards achieving the goal.

Mathematical formulas also apply to Web application's user interfaces as show by different usability studies. Out of all of them, the most popular is the Golden Ratio rule, based on Fibonacci sequence, which can be applied to interface's structure, font size and more [30].

Web development also requires extensive server-side and client-side optimization, from images, text, page sizes, to caching, database server performance, Web server security and more.

Web applications and Web sites can be developed using a huge number of technologies and programming languages. There are three main areas in Web development: client-side technologies, server-side technologies and database technologies.

In order for Web pages of a site or Web application to be displayed in the browser of the user, the browser must be able to interpret the markup language the page is provided in. Such markup languages include HTML, XHTML, which is a combination of HTML and XML (Extensible Markup Language), HTML5. All of these can be styled using cascading style sheets (CSS). Additional functionalities can be added to Web pages by using JavaScript, Flash movies or applications, JavaFX rich Internet applications, Ajax requests, or enhancements added using Microsoft's Silverlight.

HyperText Markup Language (HTML) was invented by Tim Berners-Lee, the inventor of the World Wide Web. HTML is used to create structured documents. It has proper structural semantics for text ranging from heading, to paragraphs, links, quotes or lists. It also allows documents, images, object embedding to enhance the content of a Web page. HTML is also used to create interactive Web forms. However, these forms must be processed by a server-side technology as HTML itself doesn't possess such features [5].

JavaScript can also be embedded into HTML pages to enhance it. JavaScript is an implementation of ECMAScript and is implemented as part of a Web browser and can be switched off from its preferences panel. JavaScript is mainly user to provide rich user interfaces and to add dynamism to Web sites [30].

Cascading Style Sheets are used to style and change the appearance of Web pages. It was designed to enable the separation between the content of a Web page and its design, or document presentation. Rules available in CSS can be applied to change layouts, colors, fonts, transitions and more. Separation of content from design offers more flexibility and control inside development teams [31].

Adobe flash (previously Macromedia flash) is a multimedia platform that can be used to add animations, videos and interactivity to Web pages. It can be used to develop rich Internet application which resemble desktop applications, but is mainly used for the development of online games and advertisements. Flash applications can be programmed using its built in object oriented language: ActionScript. Flash can manipulate vector and raster images, text, drawings etc. It can also capture various input devices [32].

However, flash requires a lot of memory and some mobile devices, such as Apple's iPhone, don't support the technology, and according to Apple's CEO they won't do so in the future [33]. Since the press release many Web experts have said that with the growth of mobile device market share over Web browsing, Adobe's Flash platform is likely to become obsolete in short time.

JavaFX is a java platform that can be used to develop rich Internet applications and is based on Java, and is available on a large number of platforms, Windows, Linux, Mac being included [34].

Silverlight is a Web application framework that has similar functionalities to those of Adobe's Flash platform. It is compatible with multiple Web browsers, and mobile devices are likely to be supported in the near future [35].

Server-side technologies can be grouped into two categories: programming or scripting languages and frameworks. The most popular programming and scripting languages currently in use are PHP (Hypertext Preprocessor), ASP (Active Server Pages), Python, Ruby, Groovy, Perl or ColdFusion, while the most popular frameworks are CakePHP (PHP), Django (Python), RubyOnRails (Ruby) and GRails (Groovy).

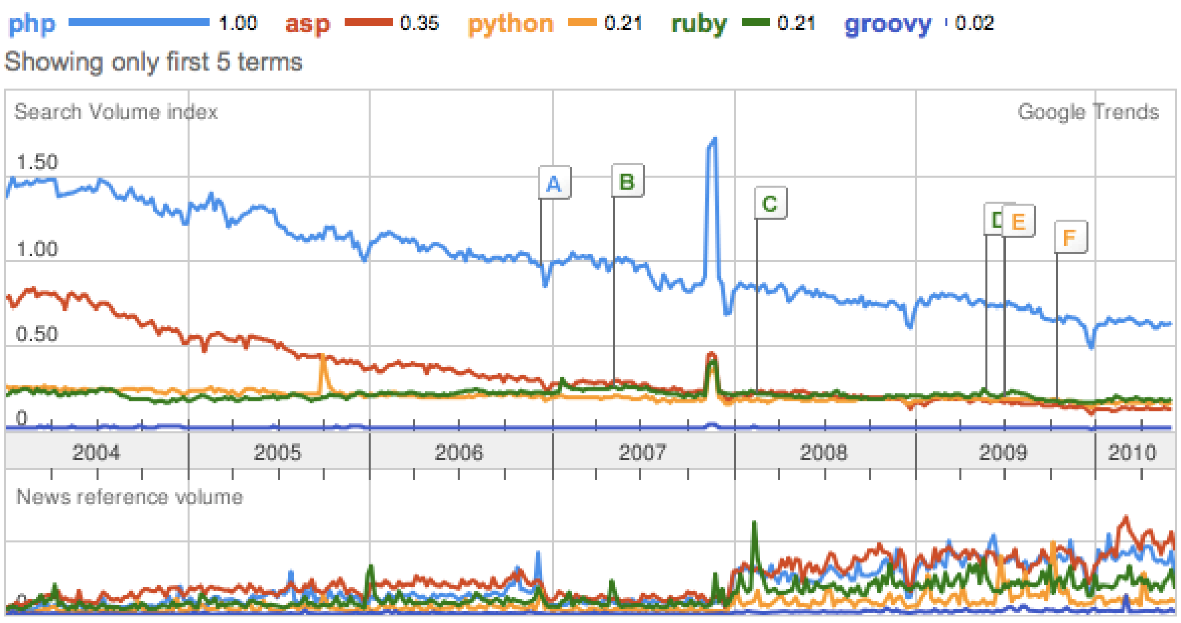


Figure 3. Google Trends statistic for PHP, ASP, Python, Ruby and Groovy as of 18.06.10

According to Google Trends (Figure 3) PHP, ASP and Ruby are the most popular programming or scripting languages. Also, according to same Google Trends statistics, Django is by far the most popular framework, with CakePHP coming into second place.

PHP is a recursive acronym for HyperText Preprocessor. It is widely used scripting language suited for Web development which can be embedded into HTML. By using PHP dynamic Web applications and Web sites can be developed. PHP files are processed by an interpreter application in command-line mode performing desired operating system operations and producing program output on its standard output channel. Therefore, PHP files do not require compilation [36].

ASP (Active Server Pages) was Microsoft's first server-side scripting engine. ASP can be used to generate dynamic Web pages. It was released as add-on to IIS server. Since 2002, ASP was surpassed by ASP.Net which is a Web application framework which allows programmers to build dynamic Websites, Web applications and Web services [37].

Ruby is a dynamic programming language which has a complex grammar and a core class library with a rich and powerful API. Ruby was inspired from Lisp, Smalltalk – just as Apple's Objective-C – and Perl. However, Ruby uses a grammar that is easy for C and Java programmers to learn. Ruby is a pure object-oriented programming language [38].

"Ruby is designed to make programmers happy" – Yukihiro Matsumoto, creator of Ruby

Django is an open source framework written in Python which follows the MVC (Model-View-Controller) pattern. Django eases the creation of complex, database-driven Websites and Web applications. It offers CRUD (Create, Read, Update, Delete) interfaces which are dynamically generated on runtime, just as RubyOnRails' dynamic scaffolding. Django emphasizes reusability of components and rapid development [39].

CakePHP is an open source framework written in PHP which also follows the MVC pattern, just as Django does. CakePHP makes it easier for the user to interact with the database using an active record pattern, just like RubyOnRails [40].

Each of these server-side technologies allows database interactions, and each programming or scripting language is best suited with one or more database technologies, such as MsSQL, MySQL, PostgreSQL, SQLite or Oracle.

MsSQL, MySQL, PostgreSQL and SQLite are all based on Structured Query Language (SQL). SQL is a database computer language that can be used to manage data in a relational database management system. It can do data queries and updates, schema creations and modifications and data access control [41].

MsSQL is a relational model database server developed by Microsoft which implements T-SQL and ANSI SQL query languages. MySQL is similar to MsSQL, but it is a GNU General Public License solution. MySQL is owned by Sun Microsystems, subsidiary of Oracle, which also develops the relational database management system with the same name. PostgreSQL is different from the rest of the database technologies presented because it is an object-relational database management system (ORDBMS). Finally, SQLite is an embedded relational database management system contained in a relatively small C programming library. SQLite is different from the rest of the database systems presented because it doesn't run as a single standalone process with which user developed applications communicate. SQLite library is linked in and therefore becomes part of the software that's being developed using this technology [42].

2.3 Technologies and Concepts Used in iPhone Development

Web development and desktop software development may be closely related, especially if comparing Web applications or huge dynamic Web sites with desktop software. The similarities may not go as far as static Web sites, but for complex applications, desktop or Web, certain steps must be taken into consideration and shouldn't be skipped.

However, when developing for mobile phones, especially smartphones, specifically touch devices, everything changes. The development process is basically the same with software development, but the similarities end here.

For starters, users of a Web application or desktop software may stop and use a given tool for an extended period of time. Take for example writing tools, games or financial related software. Once opened, it is likely that the users will use them for at least a couple of minutes, if not hours.

On the other hand, research shows that the iPhone usage pattern is radically different: "Fast launch, short use". Typical iPhone users pull their device out of their pockets and use it for a few seconds, maybe a couple of minutes, before putting it away again. During this time, the user may be making a phone call, search for a contact, change a song in their music player, or use a third party application [43].

Having such a short period of time to getting things done, the user expects the application to have a simple, minimal user interface that doesn't get in their way, but instead helps them getting the job done faster. Therefore, iPhone applications must be developed to make use of this usage pattern. If the user interface is not specifically designed for the device, and just ported from a Web or desktop application, it will certainly fail. To conclude, the user interface, look and feel of a touch device application has to be well designed.

Secondly, on iPhone OS, there are no background applications. This means that only one application runs at a given moment in time. As users cannot switch between applications as some of them are loading, it is likely that the user will not use a specific application if its loading process takes too much time. Moreover, the application must be prepared to quickly exit. Whenever one user leaves the context of an application, whether by pressing the “Home” (exit) button or by using a feature that opens another application, the iPhone OS will automatically terminate the current application.

When that happens, the application should save its state and preferences made by the user on the disk and exit as quickly as possible. If this process takes longer than 5 seconds, the iPhone OS will kill the current’s application process and all unsaved data is lost [43].

Therefore, while programming an iPhone application, these auto save state features must be implemented in order to have a quality application. In terms of Web or desktop application development, such features aren’t mandatory as applications runs side by side and don’t interfere with each other.

Saving the state of the application before it quits is only half the task. A good iPhone application must also restore the state from disk when it is started. Doing so will make the application look as if it were running in the background while the user switched it for other applications. Having this feature also provides a consistent user experience by putting the user right where they were when they last used the application. This also saves time by eliminating the need to navigate from the home screen to the last screen which was in use in the application [43].

Another important aspect of a touch device application’s development is the screen real estate developers could use to insert their interface elements. If 96% of desktop and laptop computers have a resolution equal to or higher than 1024 by 768 pixels [44], the resolution on a mobile device is way lower. iPhone’s resolution is only 480 by 320 pixels (though the latest version, to be released June 2010 will have 960 by 640 pixels). Having such a small display, unnecessary features and buttons must be eliminated from the application’s design.

On the other hand, although the resolution is low, the buttons, controls and text must be big enough for users to touch or read. A perfect balance must be met. A user interface that has few features, few controls, but huge touch areas will fail as it wouldn't be usable as controls wouldn't leave space for information. A user interface that has many features, lots of controls and small touch areas would also fail as users wouldn't be able to use the application.

Another important aspect that shouldn't be neglected is the fact that memory is a critical resource in iPhone OS because its virtual memory model doesn't include swap space. Allocating more memory than what is available on the device will block the entire device. If low-memory conditions occur, iPhone OS warns and can terminate applications if problems continue. Therefore, applications must be responsive to memory usage warnings. Reducing the memory footprint of an application can be done by eliminating memory leaks, having resource files as small as possible and lazily loading the resources required to run the application [45].

Lastly, the application has to be simple an intuitive as research shows that users spend just a few seconds in trying to find or read a help guide. The usage of the application should be immediately obvious to users.

In order to develop iPhone native applications, iPhone's SDK (Software Developer Kit) provides the tools and resources needed for the task. Native applications have access to all the features of the iPhone, such as its Multi-Touch interface, accelerometers and location service. Native applications can also save data on the file system, communicate with other installed applications and more. Native applications under iPhone OS are developed using the UIKit framework that provides basic infrastructure and behavior and that makes possible to create functional applications in minutes. The framework also provides means for customization and behavior extending [43].

Developing iPhone applications can currently be done by using Apple's IDE: Xcode, which only runs on Macintosh computers, while the primary two languages used to code applications are Objective-C and Cocoa.

Objective-C is an object oriented programming language developed by Apple that adds Smalltalk message passing to C programming language. Objective-C is designed to give C full object-oriented programming capabilities [46].

Cocoa is also a object oriented programming language. It is one of the major APIs available in Mac OS X and iPhone OS. Unlike Objective-C, the Cocoa programming environment can be accessed using external tools from Python, Perl, Ruby and others by using bridging mechanisms such as PyObjC, CamelBones and RubyCocoa. For application users, Cocoa based applications usually have a distinctive feel because Cocoa automates many aspects of an application to comply with Apple's human interface guidelines [47].

iPhone’s hardware is extremely rich. It has multi-axis accelerometers, Multi-Touch screen, camera, Assisted-GPS. Latest version of the iPhone also has a digital compass and gyroscopes.

Accelerometers are devices that can measure acceleration, and acceleration relative to freefall. By doing so, these devices can detect orientation, falling or vibration shocks. By using the built in accelerometers, applications can react based on information provided by these devices, and serve as input devices for applications to change screens, view modes or to act as joysticks in games [48].

Gyroscopes are devices that measure orientation. Its different from the basic accelerometer because its ax is free to take any orientation, resulting in higher device orientation precision and smoother transitions applications can make use of [49].

A Multi-Touch screen is a screen that can detect many input touches, starting from basic single touch and going to 4, 5 or more simultaneous touches.

Assisted-GPS is a system which improves startup performance of a regular satellite-based positioning system. It uses data from surrounding cell towers to precisely determine its location, and does so in a faster time than regular GPS (Global Positioning System) systems. This is extremely important as signals from satellites are somewhat distorted inside cities, which are the most common usage locations for mobile devices [50].

iPhone’s digital compass can be used to detect device orientation relative to Earth’s north pole, provided the fact that there are no magnetic interferences around the device.

2.4 Novel Technologies in Web and Mobile Development

Geolocation is the process during which the geographic location of a user or a computing device (pc, laptop or mobile phone) is computed using a variety of data collection mechanisms. Geolocation may also refer to the process of estimating the location, not just the exact location [51].

Geolocation services use either network routing addresses or data from an internal GPS device to determine the location. Geolocation is a device-specific API (Application Programming Interface). Therefore, the assumption that every browser supports it is wrong. Some browsers support Geolocation services while others don't [52].

While the above description fits Web browsers, the case for mobile devices is different. It's true that some mobile devices have Geolocation services embedded into their mobile Web browsers. However, Geolocation services available in these browsers rely on the hardware available in the device.

Mobile devices, especially smartphones, released in the past years all have GPS devices embedded inside. Some have Assisted-GPS devices, while others don't have anything to tackle Geolocation services. Older devices could have software upgrades that would create basic Geolocation needs, based on information received from cell towers only, but the accuracy of such solutions is not agreeable.

Therefore Geolocation services for mobile devices are not new, as they have been introduced years ago. However, this is not the case for mobile browsers. Some producers including Mozilla Firefox have already implemented Geolocation services into their Web browser. Others such as Google with their Chrome Web browser have also implemented the technology. Some are working on it, such as Opera, while other companies like Microsoft haven't yet made any statements regarding the development of such features inside their browsers. Mobile Web browsers such as iPhone's Safari and Android's Web browser developed on Webkit's engine also support Geolocation [53].

There are a couple of options developers have to implement Geolocation services into their browsers. Two of them are common industry standards. One option would be World Wide

Web Consortium’s Geolocation API, which is part of HTML5 and will probably become a de-facto standard. All applications that wish to perform Geolocation should support this standard.

Another option would be available to developers by using Google Gears Geolocation API, and should definitely be used as a fallback mechanism for W3C Geolocation API. The final option, which provides only rough, imprecise location detection, is using client IP address information [54].

Each producer has chosen a different path in implementing Geolocation services into their browsers. Mozilla Firefox has implemented World Wide Web Consortium's Geolocation API Specification. Google with their Chrome Web browser have implemented the technology by using their own Gears API while Opera are working on it, the Geolocation service being available in lab build of their browser [55].

Geolocation service inside Web browsers is a novel technology. However, this is not the case for mobile devices, which have had Geolocation technologies embedded in their hardware for a couple of years now.

But combining mobile devices’ Geolocation technologies with others can result in impressive features that look more like sci-fi stories than real world software applications. A mobile device that has Geolocation services, a video camera, a compass and an accelerometer or gyroscope can be used as an augmented reality device.

Augmented reality is the process of showing real-time direct/indirect views of a real world object which augments the object with virtual computer generated information. The technology enhances one’s perception over reality [55].

Augmented reality on mobile devices usually works as an indirect view of the real world, which is captured through a camera and displayed on the device’s screen, together with information annotated on top of the video feed. This way, information about user’s surroundings becomes interactive and digitally usable. Artificial information about the environment and objects within could be retrieved from a database and displayed as an information layer on top of the real world view. Advanced research on augmented reality includes use of head-mounted displays and virtual retinal displays for visualization purposes [56].

A popular augmented reality application available for iPhone users in London is “Nearest Tube”. Nearest tube uses iPhone’s GPS, accelerometer, camera and compass to offer real-time information regarding subway tube stations in London, as shown in Figure 4.

When the application starts, user’s location is detected. If the user is holding the device flat (parallel to the ground) the application shows arrows towards the directions of the nearest tube stations on all of London’s 13 lines.

If the device is tilted upwards (perpendicular to the ground), the application will show nearest tube stations, providing the direction they are in relation to user’s location, how many miles away they are and what tube lines they are on. Figure 4 shows the application with the device tilted upwards.

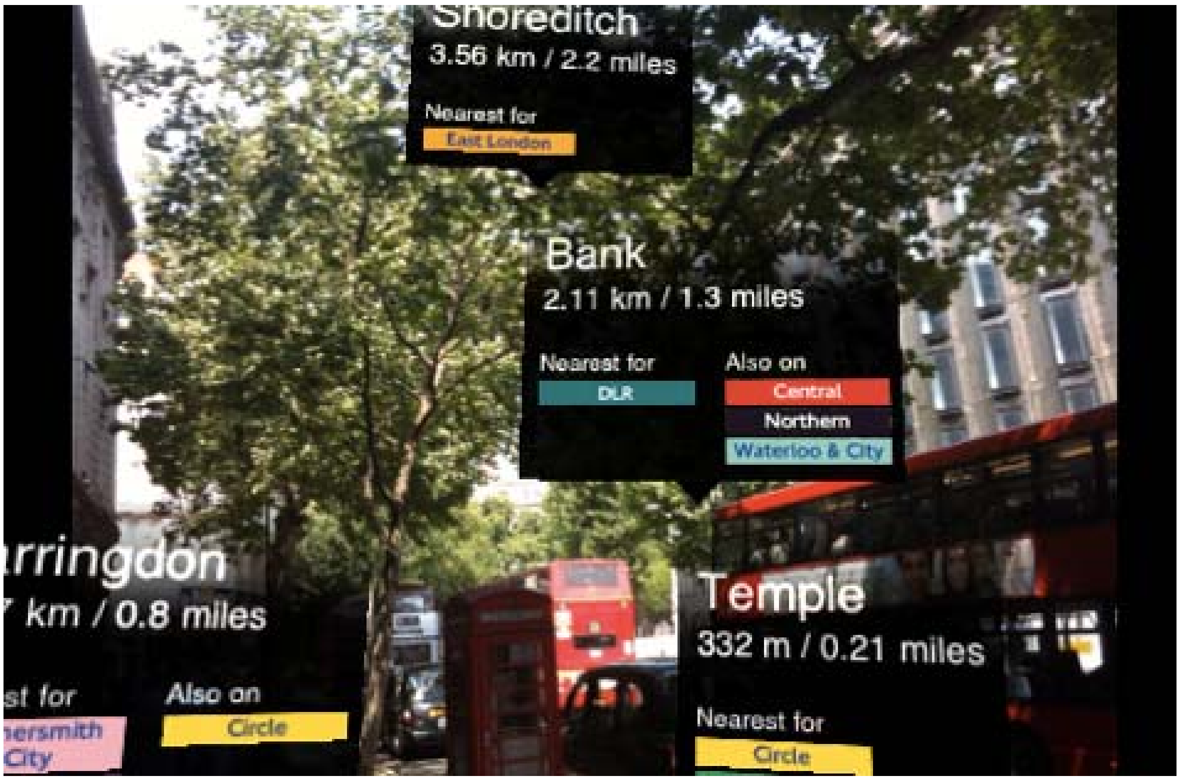


Figure 4. “Nearest Tube” iPhone application, device tilted upwards, screen capture showing live video of London and tube station information on top of it.

3. Agile Web Development with RubyOnRails

"I wanted to minimize my frustration during programming, so I want to minimize my effort in programming. That was my primary goal in designing Ruby. I want to have fun in programming myself." —Yukihiro Matsumoto (Matz), creator of Ruby [57].

"Ruby is a <<best of breed>> language that has been assembled from the best and most powerful programming features found in its predecessors." — Jim White [57]

Ruby is an object-oriented programming language that remained unknown a couple of years since it was developed in 1993 and first released as alpha in 1995. This is due to the fact that Ruby documentation was only available in Japanese, the mother tongue of its inventor: Yukihiro Matsumoto. Eventually, in 1998 Matz began to promote Ruby in English and one year later the first English official Website was launched.

The exposure of Ruby throughout the world was low. In 2000, IBM published an article with a brief overview of Ruby and an interview with its inventor. Despite Ruby's huge capabilities, it looked as if Python and PHP were going to win the race to become "the next Perl" as general scripting and Web languages. However, everything changed in 2004 when the first Ruby based tool was released by Copenhagen based student David Heinemeier Hansson. The tool was called RubyOnRails and since its release, it has become impossible to publish any book or article about Ruby without mentioning RubyOnRails [57].

RubyOnRails is a Web application framework that has propelled the popularity of Ruby outside of Japan to hundreds of thousands of developers all now interested in using the language. An important factor in the development of RubyOnRails was the Danish successful Web software company 37Signals, which in 2003 wanted to develop their own project management tool. The owners of 37Signals hired David Heinemeier Hansson to develop the application using Ruby.

The application was released four months later, in February 2004 and has had huge success. The development methodology adopted by 37Signals and Hansson was proven, and 37Signals began a rapid transition into an application development company, with Hansson eventually becoming a partner at the company [57].

Hansson used Ruby's object orientation and reflection features to build a framework that made developing database-driven Web applications easier than ever before. This framework became known as RubyOnRails and was first released to the public in July 2004. Just as Ruby, the RubyOnRails framework didn't immediately experience an explosion of popularity, but found a small number of fans who began to realize its power and began using it to develop their Web applications.

However, popular, large Web sites and complex Web applications currently in use today were developed using RubyOnRails. This includes Twitter (Microblogging Platform), Scribd (Social Reading), Hulu (Video-Streaming Platform), Shopify (E-Commerce), Github (Git Repository Hosting), Justin TV (Video-Streaming Platform), Seeking Alpha (Stock Markets) and many others [58].

RubyOnRails is agile friendly. It doesn't explicitly use different agile practices into the coding process itself. The reason is both simple and subtle. Agility is part of the fabric of the entire RubyOnRails framework. As RubyOnRails programmers get familiarized with the framework, they realize that the four preferences of the 2001 Agile Manifesto: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan, are all met by RubyOnRails [59].

3.1 Ruby programming language

Ruby is a dynamic object-oriented programming language that makes programming enjoyable and fast. It was invented by Japanese programmer Yukihiro Matsumoto in 1993, with the first public release of the language in 1995. Ruby has been adopted by programmers in Japan and has gotten documentation written in Japanese until 2001, when first English written books on Ruby were published [38].

Ruby has an easy to use interpreter, familiar syntax and powerful class libraries. Ruby draws inspiration from Lisp, Smalltalk, and Perl, but uses a grammar that is easy for C and Java programmers to learn [60].

Ruby has become a language which is applied to a wide range of fields, ranging from simple tasks to complex applications, from text processing to large-scale applications. [38]

To hundreds of thousands of programmers, Ruby has revealed a new way of thinking about programming and software development. Most developers say that Ruby is an easy to learn language but hard one to master. Ruby has surprisingly similar concepts to languages such as PHP, Perl, BASIC, C, or Pascal. However, Ruby takes a different perspective with problem solving due to its differences in terms of syntax, culture, grammar and customs. In fact, Ruby has more in common with languages such as Lisp and Smalltalk than with better-known languages such as PHP and C++ [57].

Many companies use or support Ruby in one way or another. These companies include Sun Microsystems, Intel, Microsoft, Apple, and Amazon.com. The RubyOnRails Web framework is a system for developing Web applications which uses Ruby as its base language and it powers hundreds of large Web sites.

Grammarians, biochemists, database administrators, and thousands of other professionals use Ruby to make their work easier. Ruby is a truly international language with almost unlimited applications [57].

Everything you manipulate in Ruby is an object, and the results of those manipulations are themselves objects, with no exceptions. However, many other languages make the same claim, but that isn't unfortunately the case. For instance, even data as simple as a digit is an object in Ruby, while in Java for example a digit could be an Integer object, or a non-object int value [61].

Ruby is an interpretive language, therefore no compilation is needed. Programs can be edited on the fly and sent to the interpreter. Hence Ruby provides a fast development cycle. Moreover, Ruby is also a dynamic language. Most of the tasks are done during runtime. This includes determining variable and expression types at runtime as well as determining class and method definitions. Moreover programs can be generated within programs and executed [38].

Ruby has abstract loop iterators built in, which means a block of code can be attached to a method call. For example, *Array* has the *each* method to iterate over its contents. Using this feature, developers need not worry about loop counters or boundary conditions.

Just like any advanced programming language, Ruby has support for exception handling. Therefore, the reliability of developed programs can be enhanced as they wouldn't crash so often but display human readable errors [38].

In Ruby, class and module definitions are executable code. Although parsed at compile time, the classes and modules are created at runtime, when the definition is encountered. This allows more dynamic programs than in most conventional languages [61].

Ruby is bundled with class libraries that cover a wide range of domains, from basic data types such as strings or arrays to networking and threaded programming. Moreover, huge numbers of unbundled libraries can be downloaded from rubyforge.org, Ruby's library repository, and inserted into programs [38].

Ruby, like most object-oriented programming languages do, implements garbage collection. Ruby's garbage collector recycles unused objects automatically.

Finally, Ruby is highly portable to many platforms including UNIX, DOS, Windows, Mac OS, Linux and more. Most importantly, Ruby programs run on different platforms without modification, just like Java's programs do [38].

IRB or FXRI for Windows is Ruby's interactive environment. Interactive Ruby (IRB) can be used to immediately test section of programs. Typing a single instruction into IRB and hitting the "Return" key would immediately get processed and show results [57].

3.2 RubyOnRails framework

RubyOnRails (Rails or RoR) is an open source Web application development framework written for Ruby. Its intended use is with an agile development methodology that is used by Web developers for rapid development.

Like many Web frameworks, Rails uses the Model-View-Controller (MVC) architecture pattern to organize application programming. One of its interesting features is that it imposes some fairly serious constraints on how Web applications are structured. Surprisingly, these constraints make it easier to create applications [59].

Rails includes tools that make common development tasks easier "out of the box", such as scaffolding that can automatically construct some of the models and views needed for a basic Website.

RubyOnRails includes WEBrick, a simple Ruby Web server, and Rake, a build system. Together with Rails these tools provide a basic development environment. Rails applications rely on a Web server to run them. Mongrel is usually used by most developers, but other options such as original WEBrick, Lighttpd, Abyss and many others [62].

Rails is separated into various packages including Active Record (object-relational mapping system for database access), Active Resource (Web services), Action Pack, Action Mailer and others. Prior to version 2.0, Rails also included the Action Web Service package that is now replaced by Active Resource. Developers can also create plugins to extend existing packages.

RubyOnRails' Active Record is its model support. Generally, Web applications keep their information in a relational database. Relational databases are designed around the mathematical set theory and although this is good from one point of view, it makes it difficult to combine relational databases with object-oriented programming languages [62].

Objects are all about data and operations, while databases are all about sets of values. Operations that are easy to express in relational terms are sometimes difficult to code in an object-oriented system with the reverse being also true.

There are two approaches in fixing this issue. One organizes software programs around the database while the other organizes the database around the program. Early software products that used relational databases usually had SQL queries embedded into their code and views by using strings or a preprocessor that converted SQL source into lower-level calls to the database engine [62].

The integration meant that it became natural to combine the database logic with the overall application logic. This style of programming is also common in scripting languages such as Perl and PHP today. It's also available in Ruby through its Direct Database Access Layer (DBI) library.

This approach is widely used especially in small applications. However, problems arise due to intermixing business logic and database access. The biggest problem is the process of maintaining and extending the applications.

One update to a database table could require multiple updates throughout the application's code. Developers need to go through all different sections of the software and update the code accordingly. If one section is missed that's a source of errors [62].

Object orientation and encapsulation solves these types of problems, resulting in a single place to update the code if such changes should occur. This idea has been extended to database programming. Database access is wrapped behind a layer of classes while the rest of the application uses these classes and their objects without interacting directly with the database.

This way all database schema specific problems are encapsulated into a single layer and the rest of the application is separated from low-level database access. Should any changes occur within the database, the only requirements would be to change the class that wrapped the database table.

However, real-life database tables are interconnected, and objects within an application should mirror this relation. But this results in issues around performance and data consistency. Object-relational mapping (ORM) fixes these issues and Rails uses ORM [62].

ORM libraries map database tables to classes. For example, if a database contains a table called restaurants the software will also have a class called Restaurants. Rows in the table correspond to objects of the class, and within the object attributes are used to get and set the individual columns of the database. The Restaurant object would have methods to get and set contents from and to the database [59].

Additionally, Rails classes that wrap database tables provide a set of class-level methods that perform table-level operations such as searching. Also, objects corresponding to individual rows in a table have methods that operate on that row. Therefore, an ORM layer maps tables to classes, rows to objects, and columns to attributes of those objects. Class methods are used to perform table-level operations, and instance methods perform operations on the individual rows.

The ORM layer supplied with Rails is Active Record which maps tables to classes, rows to objects, and columns to object attributes. Active Record removes the need in working with underlying database. Therefore, developers can focus on business logic part of the application.

Active Record integrates seamlessly with the rest of the Rails framework. For example, if a Web form sends the application data related to an object, Active Record can extract it into a corresponding model. Active Record also supports sophisticated validation of model data, and if the form data fails validations, Rails views can extract and format errors with just a single line of code. Active Record is the solid model foundation of the Rails MVC architecture [59].

For example, the following Rails code describes a User model which requires some basic data validation. It includes a library which is used to validate email addresses and then contains 4 different validation instructions. The first 3 validate different model attributes that can be used in Web forms and in which users may type information. The error messages which may be displayed in a Webpage if validation errors are detected are Rails' default errors based on each type of validation. The last validation instruction is similar to the first 3, but in case of an error it displays an user defined custom message.

```
class User < ActiveRecord::Base
  include RFC822 # for email validation - located inside lib folder
  validates_presence_of :username, :email, :password
  validates_uniqueness_of :email, :username
  validates_length_of :password, :in => 6..40
  validates_format_of :email, :with => EmailAddress, :message => "is
invalid."
end
```

Action Pack contains Rails' views and controllers. In a Web application, the controller supplies data to views and receives events from the pages generated by views. Because of these interactions, support for views and controllers in Rails is bundled into a single component, Action Pack. However, this doesn't mean that the code for the views and controllers are combined. Rails provides the separation required to write Web applications with clearly demarcated code for control and presentation logic.

The following code snippet presents a basic Rails application controller. The controller has a *before_filter,* an index method, and a couple protected methods. The before_filter directive ensures that when the application controller is loaded the first task to run is the set_user method, regardless of what method was requested. The other methods are used to check if the user browsing the Webpage and making requests to the application controller has required access based on requests made.

The index method is empty, and therefore when it is requested the application controller automatically loads the corresponding view.

```
class ApplicationController < ActionController::Base
  before_filter :set_user

  def index
  end

  protected
    def set_user
      @user = User.find(session[:id]) if @user.nil? && session[:id]
    end

    def login_required
      return true if @user
      access_denied("Oops. You need to login before you can view this
page.")
      return false
    end

    def admin_required
        if @user
          return true if @user.isAdmin==1 || @user.isGlobalAdmin==1
        end
        access_denied("Oops. You need admin privileges to view this
page.")
        return false
     end

    def access_denied(message)
      flash[:error] = message
      redirect_to :controller => 'users', :action => 'login'
    end

end
```

Views are responsible for creating either all or part of a page to be displayed in a browser. A view is a block of HTML code that displays some fixed text. Dynamic content created by the action method in the controller can also be embedded.

Dynamic content is generated by templates. The most common templating scheme, the rhtml (Ruby HTML), embeds snippets of Ruby code within the view's HTML using Ruby's tool called Embedded Ruby (ERb). This approach is very flexible, but violates the MVC concept because by embedding code in the view, logic that should be in the model or the controller can be inserted in the view. Maintaining a clean separation is job of the developer [59].

The following code presents a view template of the index method for the controller presented above. It is a part of a Webpage written in ERb, and embeds two Rails instructions for printing the content of two sessions.

```
<div id="home">
      <div id="home-sidebar" class="all-rounded">
            <ul id="home-switch-btns">
                  <li><a class="tl Jsearch">Search</a></li>
                  <li><a class="tr Jstumble">Stumble</a></li>
            </ul>
            <div id="home-search">
                  <label for="restaurant-name">by name:</label>
                  <input type="text" value="<%= session[:res_name] %>" />
                  <label for="restaurant-address">or by address:</label>
                  <textarea><%= session[:res_address] %></textarea>
                  <h2>and filter results</h2>
                  <label for="restaurant-price">by price range:</label>

      </div>

      <div id="home-messages" class="all-rounded">
            Initializing map...
      </div>

      <div id="map-wrapper" class="all-rounded">
            <div id="map">
            </div>
      </div>

</div>
```

The Rails controller is the logical center of a Web application, coordinating the interaction between users, views, and models. Rails controllers also have auxiliary services, being responsible for routing external requests to internal actions. Controllers also manage caching, giving performance boosts. It also handles helper modules, manages sessions and more [62].

3.3 Agile Methodologies Applied to RubyOnRails Web Development

Agile software development is a group of methodologies based on iterative development, in which software requirements and proposed solutions change during the development process due to collaboration between separate teams which have different functionalities.

In 2001, 17 programmers published *"Agile Manifesto",* a statement which presented the principles behind agile software development. These principles described a software development process in which customer satisfaction is a priority, which requires software changes as new feature requests are received from the customer. Software is delivered to the client throughout the entire process not just at the end of it. Customers and developers need to work together on a daily basis. Individuals working on the project must be motivated. Information must be provided using face-to-face conversation. Progress is measured by working software, with emphasis on good design and simplicity. Finally, the entire product must emerge from self-organizing teams which at regular intervals, reflect on how to become more effective [63].

Since the "Agile Manifesto" was published in 2001, many methods and practices have been developed to support the principles laid by the manifesto. Some of the most used agile software development methods are: agile modeling, agile unified process, extreme programming, feature driven development, dynamic systems development model, essential unified process, open unified process or scrum, while some of the commonly used agile practices are: test driven development, behavior driven development, code refactoring, pair programming or planning poker [64].

For example, extreme programming should improve software quality and the responsiveness towards changing customer requirements. Extreme programming requires frequent software releases in short development cycles. The purpose of this scenario is to improve productivity and provide milestones when new customer requirements can be introduced to the product development [65].

A couple of the practices required by extreme programming are pair programming, unit testing, code reviewing, sprints and avoiding implementation of additional features until they're actually required.

Pair programming is a technique in which two programmers work together at a single computer station. Each has a specific role, and these two roles are switched every 30 minutes. The primary role is that of the driver, the programmer that actually writes the code. The second programmer is called an observer and has to review the code while the driver is generating it. The observer must also present ideas for improvements and try to anticipate future problems. This was the driver can concentrate on completing the current task, while the observer serves as a guide and safety net.

Programmers working in pair write short programs that usually have fewer bugs and improved design. While working in pairs, more designs alternatives are considered than programmers working on their own, or freelancers. Task milestones are also reached faster in pair programming than in single programming cases.

Although tasks are completed faster, actual billable time increases in pair programming. Project managers must therefore balance faster completion times with higher costs. Pair programming should be used on difficult tasks instead of simple ones as for the later pairing often results in a productivity drop [66].

Pair programming is also beneficial when new programmers are hired, as they could work in pairs with experienced programmers. By doing so, they pick up techniques from each other, especially the new team member which learns techniques used in a specific company. Moreover, programmers working in pairs spend less time on non productive activities such as Web surfing or personal email. The ultimate benefit of pair programming would be a greater confidence over code correctness.

Another agile technique is scrum, which is an iterative, incremental framework for project management. Scrum was intended for management of software development projects, but it can also be used to run software maintenance teams.

Scrum contains a set of practices and three predefined roles. The project manager is usually the scrum master and maintains the processes. There's also the product owner, or someone who represents the client, and a team of developers.

Scrum uses sprint periods of two to four weeks during which a potentially releasable product increment is developed. The features that are implemented during a sprint are chosen from a set of high level requirements needed to be done. Which of these requirements go into a sprint is decided at the beginning of the phase.

The product owner, or client, informs the development team on the items in the requirements list which must be completed first. The team than estimates how many of these items can be completed during the next sprint. An important issue with scrum is that, during a sprint, requirements of that specific sprint cannot be changed.

A key principle of scrum is its recognition that during a project the customers can change their minds about what they want or need. Therefore, scrum focuses instead on maximizing the team's ability to deliver quickly and respond to emerging requirements [67].

For programmers working alone, scrum technique can be adapted to a one man team. The starting point for implementing scrums is to develop a prioritized list of outstanding client requirements or needed features, having for each item in the list a priority, a description, and a developing estimate. Next step would be defining a one week sprint, and start developing items in its list, while making sure that distractions aren't sidetracking the development task. The success of each sprint depends on the ability of the developer to estimate development time required for each task and stay on track [68].

A technique believed to improve scrum adapted for one man teams would be the pomodoro technique. It is a time management method developed in late 1980s. The technique uses a timer to define work periods of 25 minutes followed by short breaks. Each period is called a pomodoro (Italian for tomato). The technique is based on the concept that frequent breaks can improve mental agility. Five basic steps are required to implement this technique, in the following order: decide on task to be done, set the pomodoro timer to 25 minutes, work on the task until the timer rings, take a short break and continue with the third step or the first step if the task is completed. It is also recommended that a longer break is taken every four pomodoros [69].

This technique blends the one man team scrum as the first step – that of choosing a task – is embedding the sprint list of the scrum methodology.

Moreover, the pomodoro technique is related to concepts such as timeboxing and iterative and incremental development, which are all characteristics of agile development. The pomodoro technique has been adopted in pair programming, as programmers should switch their roles in not more than 30 minutes.

RubyOnRails is agile. However, RubyOnRails developers do not need to master agile software development methodologies and practices for them to agilely develop Web applications, because agility is part of the fabric of the entire RubyOnRails framework.

As RubyOnRails programmers get familiarized with the framework, they realize that the four preferences of the 2001 Agile Manifesto: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan, are all met by RubyOnRails [59].

RubyOnRails is all about individuals and interactions between them. There are no heavy toolsets, complex configurations or elaborate processes. There are only tight groups of developers, using their favorite IDEs and editors, and Ruby blocks of code. The framework allows developers to deploy basic CRUD functionalities in minutes; therefore the customers can rapidly see different functionalities as they're built into their solution. This process is a completely interactive process, feedback flowing rapidly from customers to developers.

Moreover, as pieces of a Web application are developed and reviewed by customers, there's little time for developers to create extensive documentation. Instead, they focus on fixing problems that arise which result in working software parts being available early in the development process. Furthermore, throughout this entire process the customer is kept close to development team, so customer collaboration is encouraged.

Finally, customer feedback not only points errors, flaws and bugs in the application that's being developed, but it also determines new features and excludes existing ones. During the development process of RubyOnRails applications requirements change and developers have the ability to respond to these changes and update sections of code. RubyOnRails is also bundled with unit and functional testing, giving developers a lot of safety when the write code or make changes to it [59].

3.4 Using External Web Services in RubyOnRails Web Applications

Rails’ Action Web Service (AWS) provides support for the Simple Object Access Protocol (SOAP) and XML-RPC (XML-Remote Procedure Call) protocols in Rails applications. It converts incoming method invocation requests into method calls on the Web application’s services and takes care of sending back the responses. By doing so, Action Web Service allows developers to focus on writing application specific methods instead of methods that handle Web service requests. However, AWS only implements basic functionalities available in the SOAP and XML-RPC specifications [59].

Moreover, with the advent of REST support in Rails since the restful_rails plugin has been merged to Rails’ core in 2007, the developers behind Rails framework have become less interested in XML-RPC–based and SOAP-based Web services. Furthermore, Action Web Service was removed from the Rails core and made into a plugin since Rails 2.0 [70].

Representational State Transfer (REST) is a software architecture for distributed systems such as the World Wide Web. Conforming to the REST constraints is referred to as being ‘RESTful’. REST was introduced and defined in 2000 [71].

Since version 2.0 when Action Web Service was removed from Rails core, Rails offers by default both HTML and XML as output formats, the latter being required for RESTful Web services [62].

The basic idea behind REST is that developers don't have to make Web services act as regular method calls or function calls. The four basic HTTP operations: GET, PUT, POST, and DELETE correspond to the four basic SQL operations: SELECT, UPDATE, INSERT, and DELETE. Therefore, it's possible to build complex applications by doing nothing more than using HTTP requests to move XML documents.

REST is a collection of resources, with three defined aspects: the base URI for the Web service, the MIME (Multipurpose Internet Mail Extensions) type of the data supported by the Web service which is often JSON, XML or YAML (YAML Ain't Markup Language) but can be any other valid MIME type, and the set of operations supported by the Web service using HTTP methods (POST, GET, PUT or DELETE) [72].

Rails is agile, and developing Web applications that have a well known features list is done really fast, especially in teams that use agile software development methodologies in their development process. Moreover, Rails makes it easy for developers to extend and enhance the applications being developed by making use of Rails' numerous plugins. Functionalities that would take hours to develop can be produced in a matter of minutes by simply installing and configuring a Rails plugin [59].

By combining this Rails plugin mechanism with RESTful Web services (by using RESTful plugins) developers could enhance their application and rapidly add features to it really easy.

For example, in a Web application that publicly displays information about its members, such as Facebook, Tumblr or Twitter, a feature that is usually required is to have avatars for each user. Implementing this feature from scratch is somewhat complex. The developer should not only handle image upload to server, but should make sure no images are overwritten during simultaneous upload processes. Images should be processed, optimized and resized to make sure the overall appearance of the Website isn't affected. Building these features and testing them would require at least half of day with tradition Rails programming. If using PHP or other languages it may take even more. However, by using an external Web service such as Gravatar, and developing functionality around it using Rails' gravatar plugin, the issue described above can be implemented and tested in as little as 15 minutes.

Another example would be implementing Twitter trends inside the page of a Web application. By using JavaScript one would require at least 50 lines of code. Instead, by using Rails' twitter plugin two lines of code are required. The first is for installing Twitter's plugin and the second firing it up to display information in a view.

Implementing a yahoo search page inside a Web application could also be completed quickly with as little as 20 lines of code as shown in [72].

Displaying images from Flickr is also a short task. Once a Rails flickr plugin is installed, little configuration is required and developers could start building functionalities into their views and have images from Flickr accounts displayed on Web applications in less than one hour's time.

4. Pros and Cons for RubyOnRails

RubyOnRails is an agile friendly framework written for Ruby. Its intended use is with an agile development methodology that can be used by Web developers for rapid development.

Rails doesn't explicitly use different agile practices into the coding process itself because agility is part of the fabric of the entire framework. The four preferences of the 2001 Agile Manifesto: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan, are all met by RubyOnRails, which sets Rails apart from other Web development frameworks.

4.1 Ruby vs PHP vs Python

Ruby is a dynamic, pure object-oriented programming language that makes programming enjoyable and fast. On the other hand PHP is a scripting language, and object-orientation was added only to its 5th version. Of course, objects could have been created since version 3 but they only acted as structures to hold attributes, as they didn't have any methods available. Python is a programming language much like Ruby, but it doesn't support only object-oriented paradigm, but many others as well, including functional programming [73,74].

Out of the three, PHP is the most popular, the other two sharing the second place, as they appear to be equally popular, according to Google Trends statistic presented in Figure 5.

However, Ruby seems to be referenced in more news than Python. This is due to the fact that in the last years a lot of Web applications developed using RubyOnRails have been released and became popular.

Python was designed to emphasize programmer productivity and code readability. Python requires correct indentation as indentation has semantic meaning, with successively larger indentation meaning tighter scopes.

PHP was designed to ease dynamic Web pages development and is very similar in terms of syntax with C and C++, preserving its brackets structures and many other features. Syntax wise, Ruby is similar to Python as brackets are used only in extraordinary situations. However, indentation is not required, but new lines are mandatory in different constructs. Ruby code is organized into blocks, with blocks starting with various constructs and ending with reserved keyword *end*.

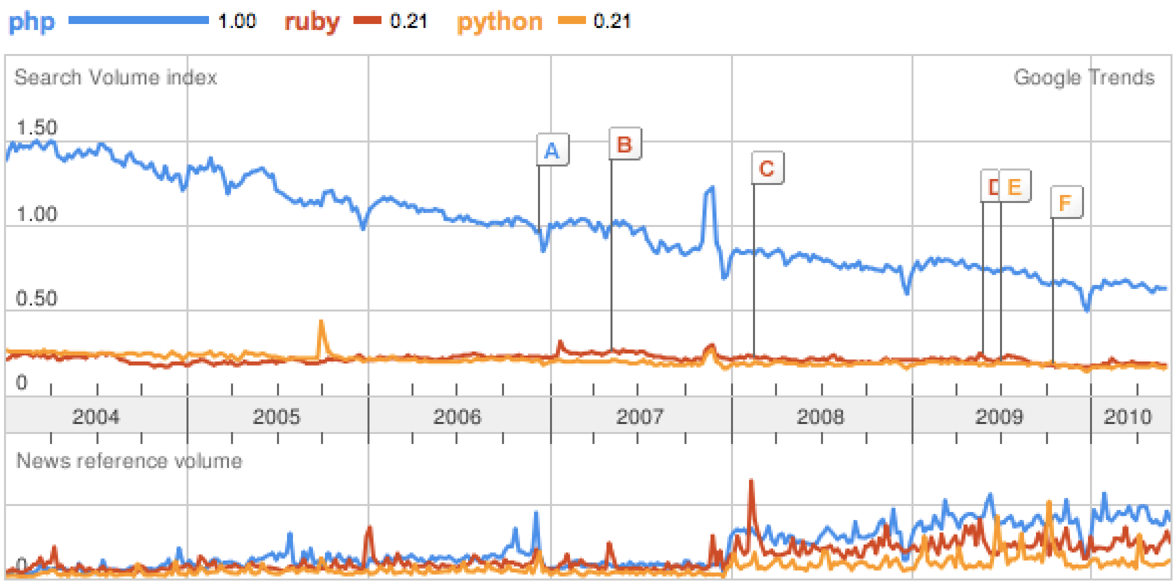


Figure 5. Google Trends statistic for PHP, Ruby and Python as of 20.06.10

Ruby and PHP are similar as their code doesn’t require compilation. Both of them are pure interpreted languages. In contrast, Python code is automatically compiled to byte code by its interpreter, much like what happens with Java [60,73,74].

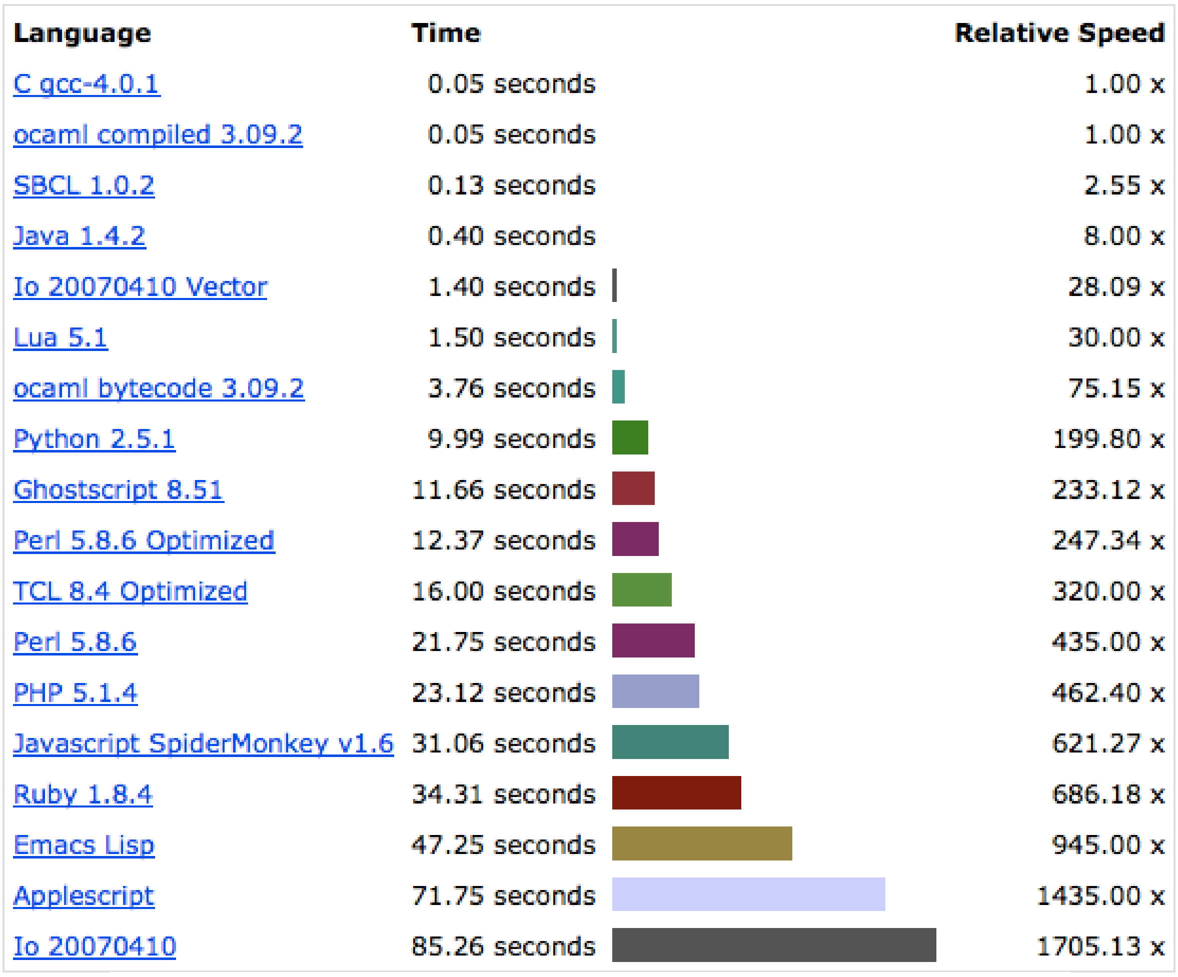
All three languages have similar exception handling mechanisms, the only difference being the reserved keywords each one uses for the process.

In terms of speed, tests showed that Python (version 2.5.1) is the fastest, while Ruby (version 1.8.4) is the slowest, with PHP (version 5.1.4) somewhere in between. However, Ruby's latest stable release 1.9.1 is much faster, and drastic improvements in terms of speed are expected with the release of Ruby 2.0 [75]. Figure 6 shows the compared results of a fractal test between these languages and others.

Without any optimizations, it seems that PHP is at least twice as slow as Python (2.31 times slower), and Ruby is more than three times slower than Python (3.43 times slower).

However, incremental loops tests show different results, with Ruby and Python leading well ahead of PHP, both in terms of execution time and memory usage. During such a test, Ruby proved to be only 50% slower than Python, while PHP was almost 4 times slower than Python. In terms of memory usage Ruby had the lowest memory footprint, Python used half as much memory as Ruby, while PHP used 4 times more memory than Ruby [76].

Depending on requirements, it looks like Python and Ruby are the ones most suitable for development. PHP has it benefits but overall can be considered a slow, resource intensive language.



Language	Time	Relative Speed
C gcc-4.0.1	0.05 seconds	1.00 x
ocaml compiled 3.09.2	0.05 seconds	1.00 x
SBCL 1.0.2	0.13 seconds	2.55 x
Java 1.4.2	0.40 seconds	8.00 x
Io 20070410 Vector	1.40 seconds	28.09 x
Lua 5.1	1.50 seconds	30.00 x
ocaml bytecode 3.09.2	3.76 seconds	75.15 x
Python 2.5.1	9.99 seconds	199.80 x
Ghostscript 8.51	11.66 seconds	233.12 x
Perl 5.8.6 Optimized	12.37 seconds	247.34 x
TCL 8.4 Optimized	16.00 seconds	320.00 x
Perl 5.8.6	21.75 seconds	435.00 x
PHP 5.1.4	23.12 seconds	462.40 x
Javascript SpiderMonkey v1.6	31.06 seconds	621.27 x
Ruby 1.8.4	34.31 seconds	686.18 x
Emacs Lisp	47.25 seconds	945.00 x
Applescript	71.75 seconds	1435.00 x
Io 20070410	85.26 seconds	1705.13 x

Figure 6. Fractal benchmark test results, timestretch.com

4.2 RubyOnRails vs CakePHP vs Django

RubyOnRails (Rails or RoR) is an open source Web application development framework written for Ruby. Its intended use is with an agile development methodology that is used by Web developers for rapid Web applications development. Like many Web frameworks, Rails uses the Model-View-Controller (MVC) architecture pattern to organize application programming.

Moreover, object-relational mapping (ORM) is part of Rails. Therefore, mapping database tables to application models is an easy process [59].

Django is a Web development framework written for Python. It encourages rapid development and clean design, much like Rails. Django allows Web developers to build high-performing, elegant Web applications quickly. Django also supports the object-relational mapping, allowing developers to easily define their models in Python, models which are mapped automatically to the database [39].

Strangely, Django doesn't fully support the model-view-controller pattern. Django's developers actually shifted the classic controller to the *view*, while the view became a *template*. Django is more a model-template-view (MTV) framework than a MVC framework.

In Django, a view describes which data is presented in the browser, not how it is presented. The template is responsible for the looks of the application, not the view. Controllers in Django are missing, because the entire framework is considered a controller. This is due to the fact that the framework sends requests to appropriate views, according to Django's URL configuration [39].

CakePHP is a Web development framework for PHP which provides functionalities for developing, maintaining and deploying Web applications. CakePHP uses design patterns like the model-view-controller and object-relational mapping. It reduces development costs and allows developers to achieve complex functionalities with reduced code [40].

In terms of application testing Rails offers its developers means to do unit, functional and integration tests. Django offers unit testing while CakePHP is bundled with functionalities to build unit tests, object mocking, fixtures, code coverage, memory analysis using SimpleTest and XDebug. Therefore, in terms of application testing, CakePHP is by far the most advanced [39,40,59].

All three frameworks have at least a basic database migration framework and all support caching and form validation. Django and CakePHP also have security frameworks based on access control lists (ACL) while Rails has numerous plugins to cope with such issues. Rails and CakePHP offer various ways for templating while Django uses its unique Django template language which is similar to PHP's Smarty template system.

In terms of JavaScript frameworks, CakePHP has adopted and implemented a large number of frameworks such as Prototype, script.aculo.us, jQuery, jQuery UI and MooTools. Rails is missing jQuery UI and MooTools while only uses jQuery in its admin sections.

Figure 7 displays a comparison between these frameworks. They were initially released in 2005, and all of them have been well received by the Web development community, each of them being adopted by an important number of developers. RubyOnRails is mostly used by Web developers using Macintosh computers, and since the Mac OS X Leopard OS released in 2007, RubyOnRails is bundled in the operating system.

Framework	RubyOnRails	CakePHP	Django
Language	Ruby	PHP	Python
MVC	YES, pure OOP	YES	NO, MTV instead
ORM	YES	YES	YES
Testing	unit, functional and integration	unit, object mocking, fixtures, code coverage, memory analysis	unit
Database Migration	YES, rake	YES	YES, south
Security	via plugins	access control lists	access control lists
Templates	YES	YES	django template language
Caching	YES	YES	YES
Form validation	YES	YES	YES
JavaScript	Prototype , script.aculo.us, jQuery	Prototype , script.aculo.us, jQuery, jQuery UI and MooTools	jQuery, admin side

Figure 7. Comparison between RubyOnRails, CakePHP and Django

5. Extending the Web with iPhone Applications

Thanks to its agility, plugins system and support for RESTful Web services the RubyOnRails framework has proven to be quite successful in deploying high quality, complex Web applications [62].

Twitter (Microblogging Platform), Scribd (Social Reading), Hulu (Video-Streaming Platform), Shopify (E-Commerce), Github (Git Repository Hosting), Justin TV (Video-Streaming Platform), Seeking Alpha (Stock Markets), and 37Signals' Web applications all proved the success of the framework.

Moreover, most of the Web applications mentioned above are extended via mobile applications on devices such as Android and iPhone. This was possible because Rails offers simple ways to create RESTful Web services, not only use external ones within the Web applications being developed.

5.1 Using RESTful Web Services in iPhone Development

Since the launch of Rails 2.0 in 2007, when REST support has been merged into Rails' core, Rails offers by default both HTML and XML as output formats, the latter being required for RESTful Web services. However, XML is not the only valid MIME type available to Rails developers. Rails also support JavaScript Object Notation (JSON), YAML (recursive acronym for "YAML Ain't Markup Language") and others [72].

Depending on usage, exporting data from a Rails application can be in done in any format: HTML for Web pages, JSON for JavaScript applications, YAML for C++, Java, .Net applications and XML for PHP, Java or .Net applications.

REST for Rails applications requires no special server setup, request encoding or decoding. Incoming requests are processed and results are served in whatever format developers choose. This is usually a custom XML format, but it isn't the sole option.

Regardless of implementation language, REST services are simple to build and easily adapt to developer's needs. Developers can also use HTTP-based filters and security measures the Web server possesses to secure their applications. Because of these benefits, and REST's simplicity, most of the Web services available on the Internet are REST services [72].

RXML templates were the first to be used when REST was implemented into Rails' core. They made the process of building REST-based servers relatively simple by using custom XML documents. In fact, to build REST-based Web servers with RubyOnRails, developers only needed to turn off standard layouts for the methods used by the Web service and associate XML templates to these methods instead of standard RHTML templates.

However, newer versions of Rails allow developers to format the result of the request directly inside the controller. For example, if an incoming request is via a Web browser, the controller responds with a HTML format, hence a Web page. If the incoming request is for JSON, and the method requested implements such a format, the result is given in JSON format [72].

This is important because a single method could handle multiple types of requests. This results in reduced code size, easier maintenance and software updates.

For example, one application that uses Rails as a backend and has its main feature a Google map containing various information should export its data using JSON format. JSON is a lightweight text-based data interchange format. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. JSON is a subset of the object literal notation of JavaScript and can therefore be used in the language without any additional requirements.

By exporting data using JSON format, information contained within JSON objects can be easily accessed through JavaScript code, which is also mandatory to handle Google Maps objects. Therefore, exporting data in JSON would make an easy blend with the front-end of such an application.

Having data exported by the Web application using JSON, developers could use it not only for above purposes, but to extend the application with mobile applications, as data would be requested by a mobile device using a JSON request and then be processed on the device and displayed to end users.

For example, iPhone's JSON-framework for Objective-C makes the process of reading and writing JSON objects a straightforward process. This way, developers could easily create CRUD functionalities through RESTful Web services provided by a Web application and remotely manipulate databases using such services.

The JSON framework for Objective-C does all the hard work of mapping the JSON structure into Cocoa data structures (NSDictionary and NSArray). Once data is inside an Objective-C object, developers can use it however they need inside a native iPhone application [77].

Moreover, developers that need to extend Rails Web applications to iPhone platforms may also use the ObjectiveResource framework. ObjectiveResource is a port of the ActiveResource framework of RubyOnRails and facilitates Rails objects serialization to and from Rails' standard RESTful Web services via either XML or JSON [78].

6. Places 2 Eat Web and iPhone Applications

Since the dawn of Web 2.0 and since the appearance of Multi-Touch smartphones there has been a lack of user oriented Web and mobile applications that support sharing of restaurant locations between people, either friends of strangers. Places 2 Eat is a system composed of two user oriented Web and mobile applications which enable users to share restaurant experiences. Both applications use modern technologies such as Geolocation and more.

6.1 Project Specification

A Web and mobile cross platform system is required. The system must enable its users to search and find restaurants suited for their needs located around themselves. The finding process must be completed in a timely manner, either by using the mobile application or the Web application. User must be able to rate and change information about restaurants inside the database, as well as adding new ones.

The mobile application will work as an extension of the Web application, connecting to the Web application to request information. Therefore, the Web application must have the most basic RESTful Web services the mobile application may use.

Web wise, users must be able to register, login, change their profile information, including their avatar or change their password. Two different administrator levels are required. The basic administrator can edit user details and ban them, create new administrators, as well as moderating information submitted by the users. Global administrators have the same abilities as regular administrator, the difference being that they can to the same tasks to regular administrators also, whereas basic administrators cannot edit or ban other regular administrators. Moreover, global administrators cannot be deleted.

Simple visitors or logged in users can browse the map and search for restaurants. They can also use filters to refine their searches. Logged in users can modify information available about a restaurant, including rating, pricing, Websites, phone numbers, emails, open hours and more.

Logged in users can also add new restaurants to the database. They first locate the restaurant, and then input information about it. Moreover, they can use Twitter and Facebook plugins to share information they've found about different restaurants.

Mobile wise, users can search for restaurants while using a full screen map. They can also filter the results much like how they do it on the Web application. They can also view details about found restaurants. Differently from the Web application, the mobile application allows users only to rate, price and add pictures to each restaurant, if they have a registered, active account on the Web application.

If the restaurant they're currently located at is not in the database, they can add it with little information required: location and name being the only requirements.

Both applications should be responsive and help user be on their way to the found restaurants. To do so, both applications should have Geolocation technologies in order to detect user's location and center the map on them. This way uses do not waste time finding themselves on the World map before searching for near-by restaurants.

Furthermore, the mobile application should also have the ability to trace user movement and update the map that's displayed on the screen as users move around neighborhoods.

6.2 Conceptual Model

The conceptual model presented in Figure 8 showcases concepts used inside the project's domain and the relationships between them. The project has two important parts. One is represented by users, who drive the community and manage the restaurant database through the two applications, and the restaurants which are the subject of users' actions. Of course, users have different access levels and actions they can make depend on these levels.

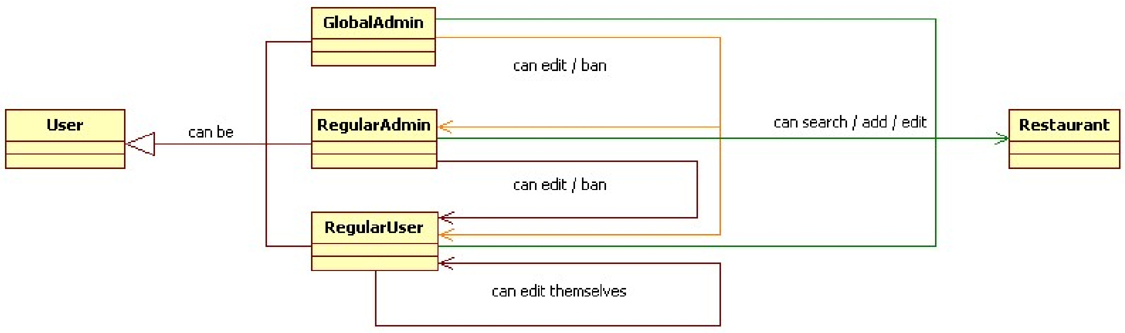


Figure 8. Conceptual model of the project

6.3 Use Cases for Web and iPhone Applications

As the Web application is complex and offers different features to its users, the use case for the Web application is split into three parts, each part for one type of user: regular user, regular admin and global admin, while the use case for the mobile application contains all possible scenarios. Figure 9 shows iPhone application use cases, while Figure 10, 11 and 12 show use cases for regular users, regular administrators and global administrators respectively. Figure 11 and 12 only show additional use cases, as they both include regular user's use cases.

6.4 MVC Architecture and Web Application's Database

RubyOnRails is a Web application framework that supports object-relational mapping (ORM). Therefore, mapping database tables to application models is an easy process, automatically done by Rails' migration framework. Since models are directly mapped to database tables, the relationship between database tables and those between application classes are practically the same. Figure 13 shows class diagrams of the Web application.

Figure 14 shows Web application's model information which maps precisely onto the Web application's database tables.

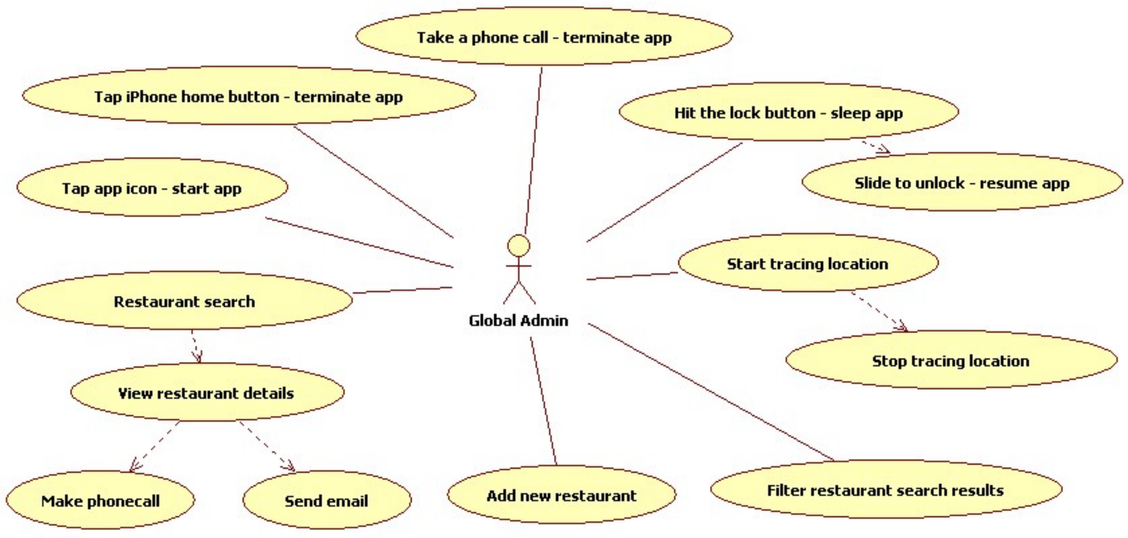

Take a phone call - terminate app
Tap iPhone home button - terminate app
Hit the lock button - sleep app
Slide to unlock - resume app
Tap app icon - start app
Start tracing location
Restaurant search
Global Admin
Stop tracing location
View restaurant details
Make phonecall
Send email
Add new restaurant
Filter restaurant search results

Figure 9. iPhone application use cases

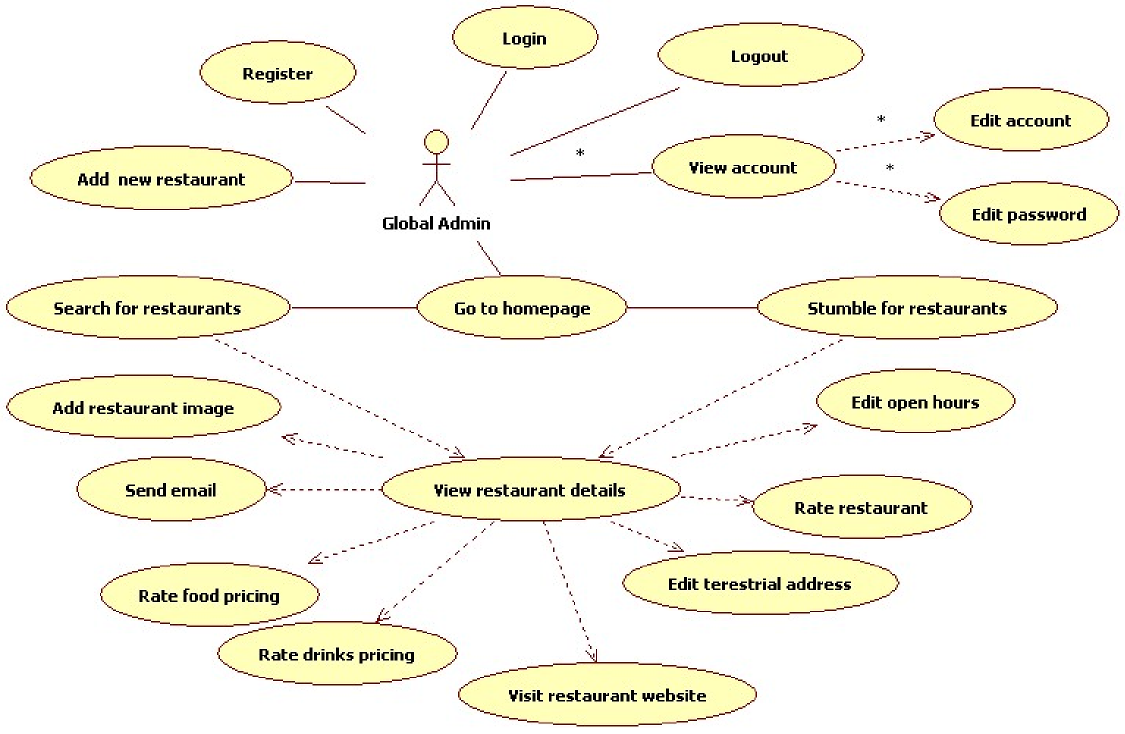


Figure 10. Web application use cases for a regular user. * - provided the user is logged in

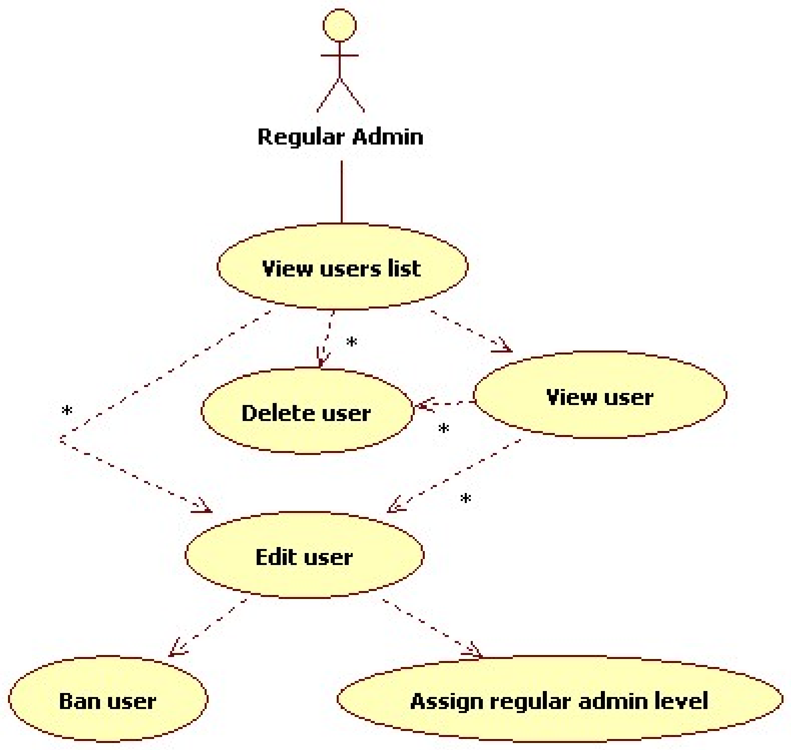


Figure 11. Regular administrator additional use cases. * - provided the user is not another administrator

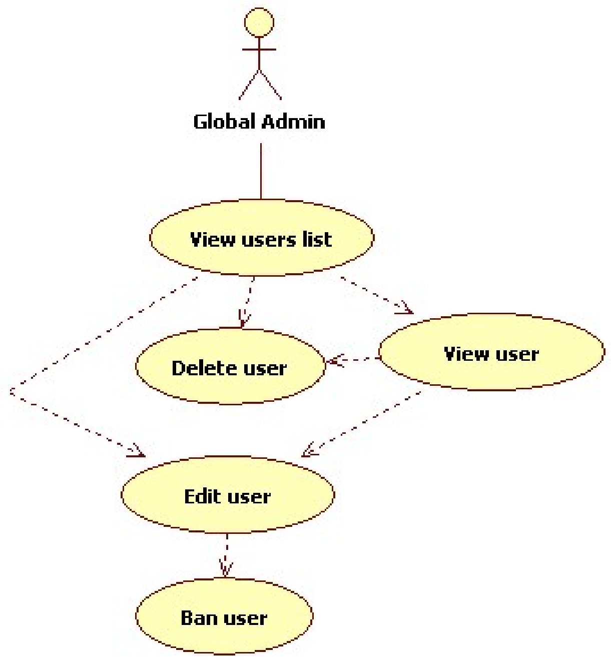


Figure 12. Global administrator additional use cases

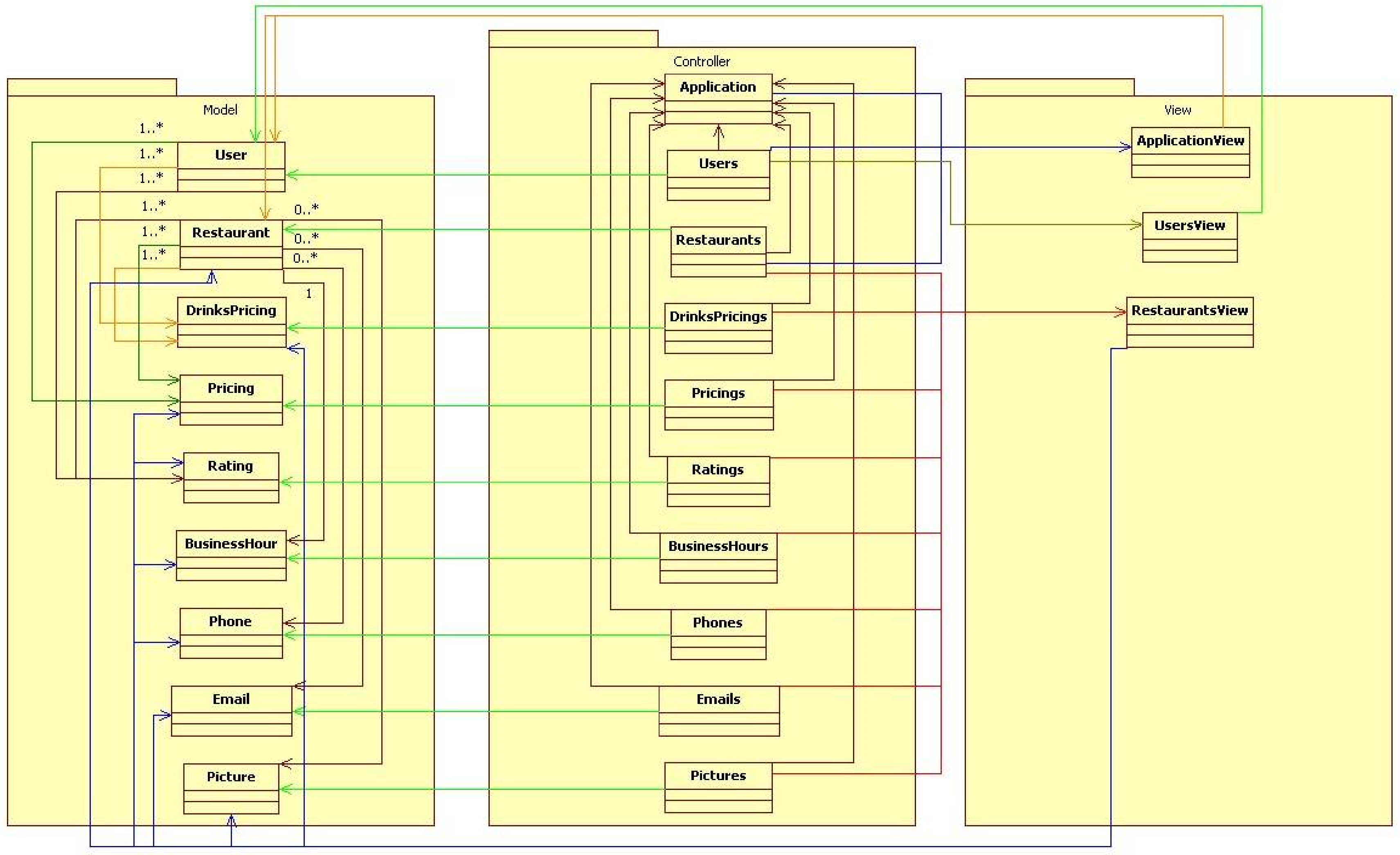


Figure 13. MVC architecture design of Web application

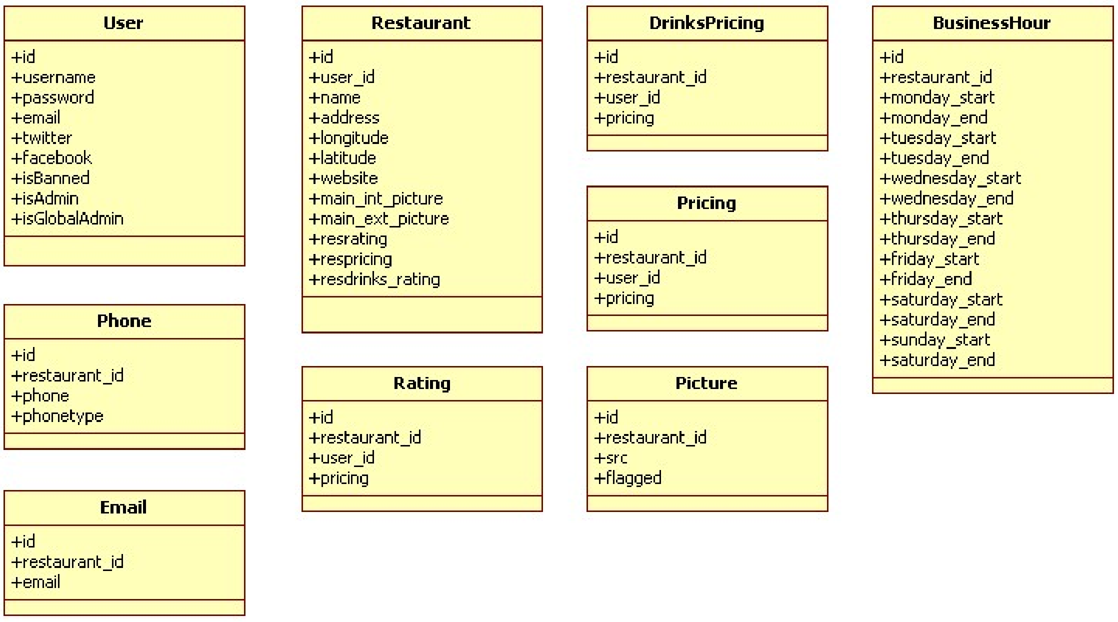


Figure 14. Web application's model

6.5 Sequence Diagrams

Figure 15 presents the sequence diagram of the register account process. In order to register a new account, users must input valid information according to application restrictions. Furthermore, even if data is valid it also has to be unique as there can be not two users with same usernames and passwords.

6.6 Key Source Code

Both the Web application and the iPhone application are quite complex. However, two key technologies are implemented in the application. One enhances the user experience while the other is extremely helpful in extending the Web application towards mobile devices or other external Websites.

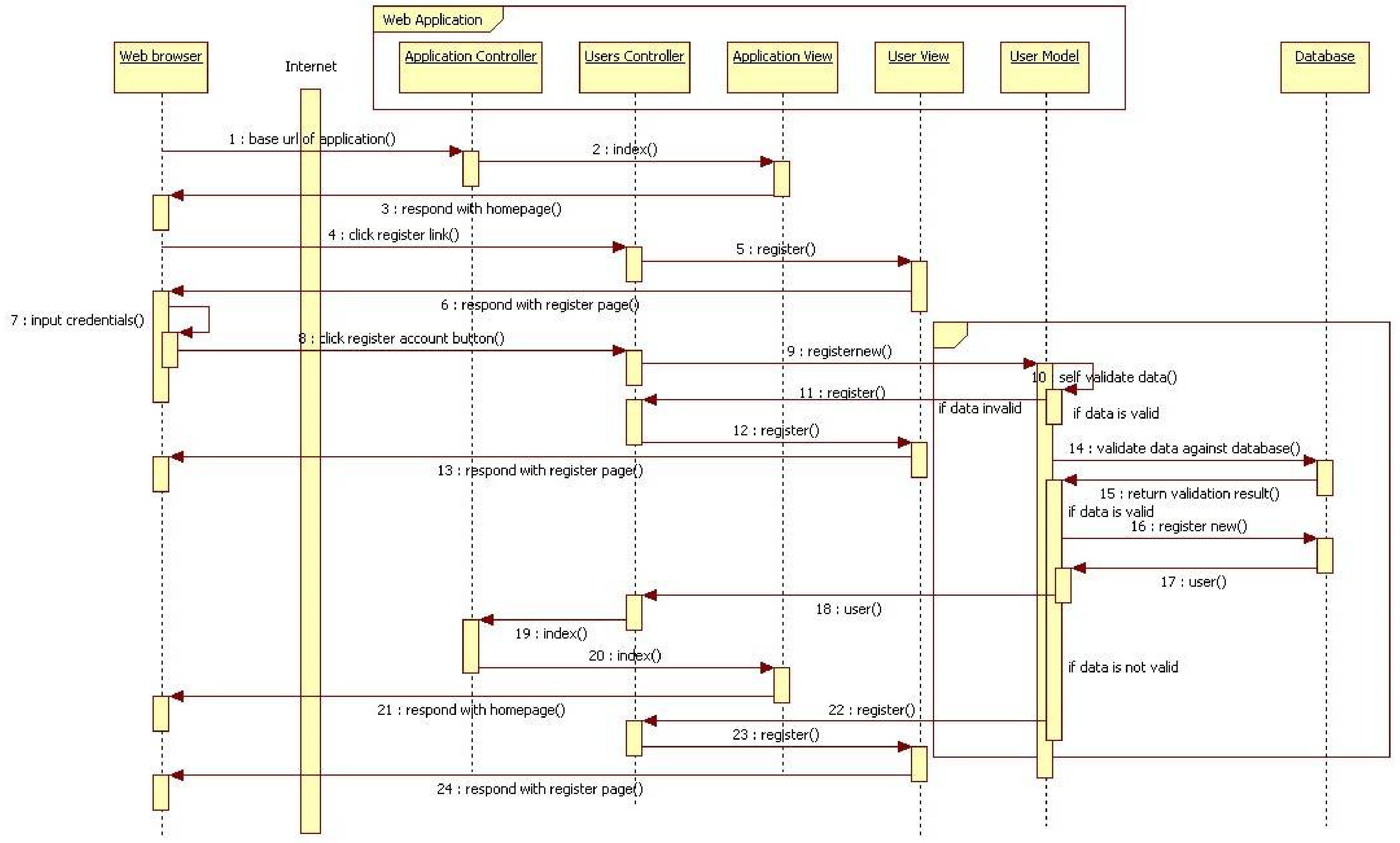


Figure 15. Sequence diagram for registering a new user account with the Web application

Geolocation services are implemented in both applications. Accessing Geolocation services on the iPhone is relatively easy, while doing the same task on the Web hasn't been implemented in many Web applications so far, and it's a new concept [79].

The following section presents a jQuery code snippet that is used to detect user location via a compatible Web browser, such as Mozilla Firefox or Google Chrome.

```
function setLocation()
{
      // detect location via W3C standard Geolocation
      if(navigator.Geolocation)
      {
          navigator.Geolocation.getCurrentPosition(function(position)
            {
                  browserDetectedLocation = new
google.maps.LatLng(position.coords.latitude,position.coords.longitude);
                  map.setCenter(browserDetectedLocation);
                  addMarker(browserDetectedLocation,'You are here!');
                  $('#home-messages').text("Location detected");
          }, function() {
            // error getting location, though supported
            $('#home-messages').text("Cannot detect location.");
          });
      } else if (google.gears)
      // detect location using Google Gears if browser supports it
      {
          var geo = google.gears.factory.create('beta.Geolocation');
          geo.getCurrentPosition(function(position)
            {
                  browserDetectedLocation = new
google.maps.LatLng(position.latitude,position.longitude);
                  map.setCenter(browserDetectedLocation);
                  addMarker(browserDetectedLocation,'You are here!');
                  $('#home-messages').text("Location detected");
          }, function() {
            // error getting location, though supported
            $('#home-messages').text("Your location cannot be found.");
          });
      }
      else
      {
            // Browser doesn't support Geolocation
            $('#home-messages').text("Your location cannot be found.");
      }
}
```

Another important aspect of the Web application is that it provides RESTful Web services mobile devices can use to get information from the Web application's database. The following RubyOnRails code snippets shows one method that can serve requests made by using a Web browser or JSON requests from a mobile device such as an iPhone. If the request is made by a Web browser and the expected result is a HTML document, the method redirects user to Web application's homepage and displays an error message as the method only serves incoming JSON requests.

```
def getpricingbounds
   respond_to do |format|
      format.html {
        flash[:error] = 'Resource not found. Error 001:format.html'
        redirect_to :controller => 'application'
      }

      @restaurants = Restaurant.find(:all)
      if @restaurants!=nil
        @restaurant = Restaurant.find(:first, :order => "respricing ASC")
        imin = @restaurant.respricing
        @restaurant = Restaurant.find(:first,:order => "respricing DESC")
        imax = @restaurant.respricing

        x = imin+0.3* (imax-imin)
        y = imin+0.7* (imax-imin)
        if session[:res_pl]== nil
          session[:res_pl] = x
        end
        if session[:res_ph]== nil
          session[:res_ph] = y
        end
        format.json  { render :json => { :pricebounds => { :x =>
session[:res_pl], :y => session[:res_ph], :imax => imax, :imin => imin }
} }
      else
        format.json  { render :json => nil }
      end

   end
end
```

6.7 User Interfaces

This section presents the key sections of the user interface of the Web application and the iPhone application. Figure 16 shows the iPhone application displaying the map of Europe. Figure 17 displays the iPhone application with the map centered on Apple's headquarters in Palo Alto.

Figure 18 shows the same application, tracing user's location, while the user is having a phone conversation. Such scenario is possible if the user turns the app after taking the call, if 3G services or better are available. As the application uses data communication, EDGE (Enhanced Data rate for GSM Evolution) wouldn't be sufficient for making a phone call while using data services [78].

Figure 19 shows the application displaying a restaurant marker which users can tap on to view restaurant name and rating, as well as a button to view a detailed screen about the restaurant.

All images display the application running in iPhone simulator on Mac OS X, hence the traced location of the user appears to be Apple's headquarters.

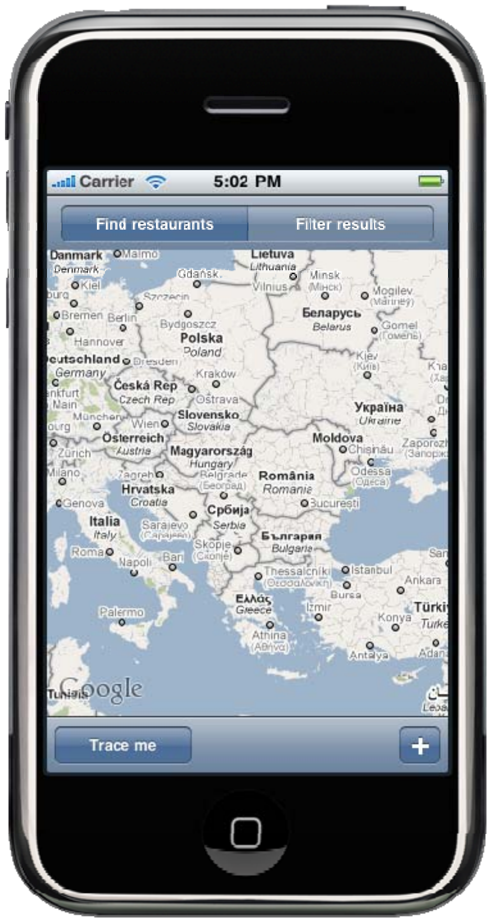


Figure 16. iPhone application displaying the map of Europe

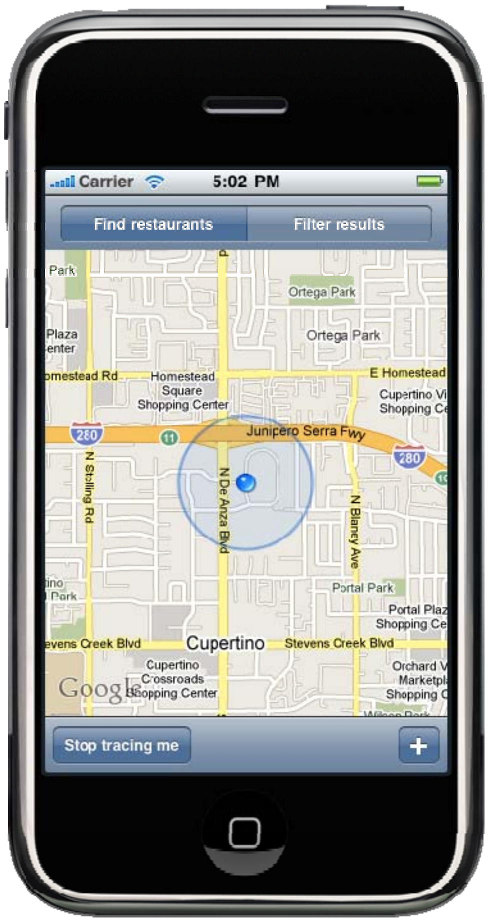


Figure 17. iPhone application displaying user's traced location

Figure 20 shows a screenshot of the Web application's homepage taken using Mozilla Firefox. It presents a first page load, therefore the Geolocation services are triggered and the user is prompted to reveal their location.

Figure 21 displays the same homepage opened using Apple's latest version of it Web browser: Safari. Version 5 was released on 7th of June 2010, and is the first version to support Geolocation services. Safari uses World Wide Web Consortium's Geolocation specification [80].

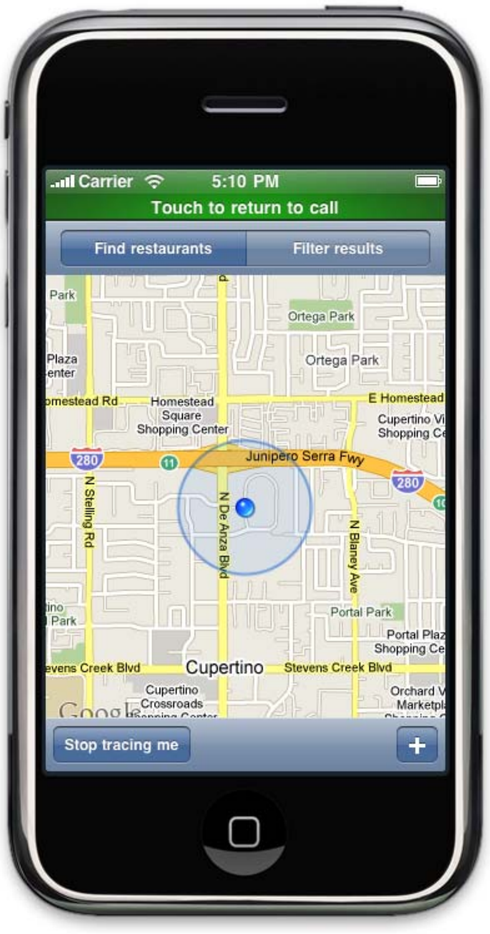


Figure 18. iPhone application in use during a phone call

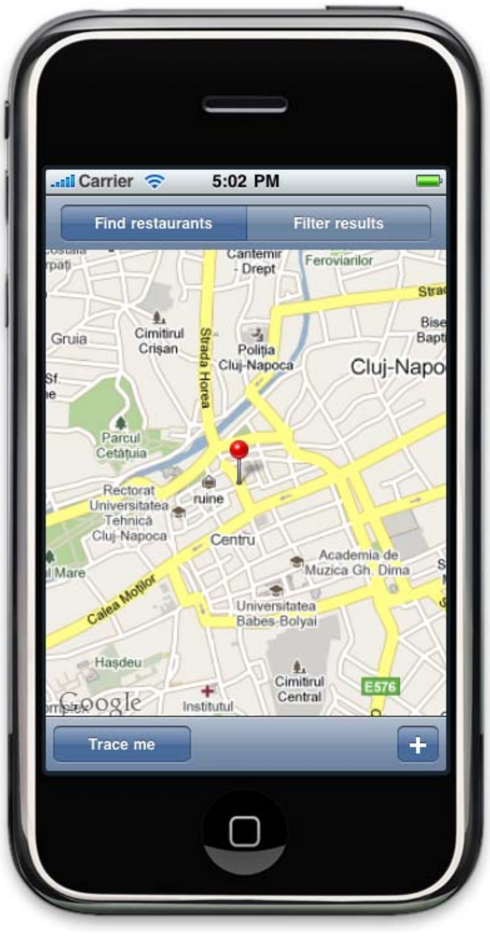
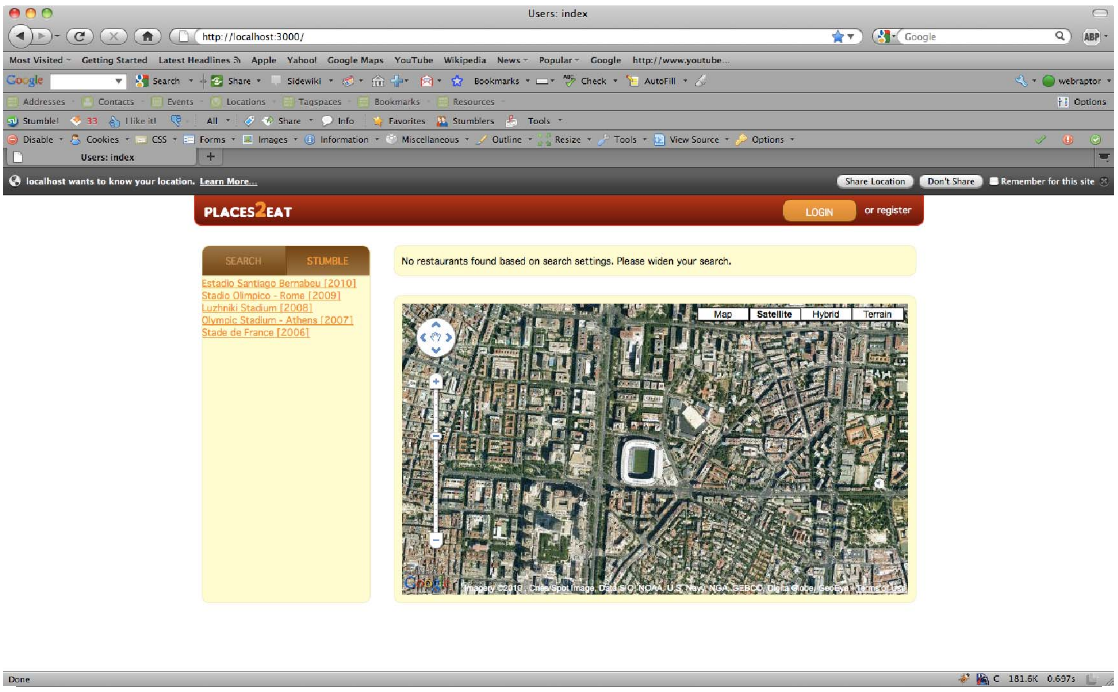


Figure 19. iPhone application displaying a restaurant marker in the center of Cluj

Figure 22 shows a login modal window of the Web application which users can use to rapidly login into the system, as page reloads are eliminated.

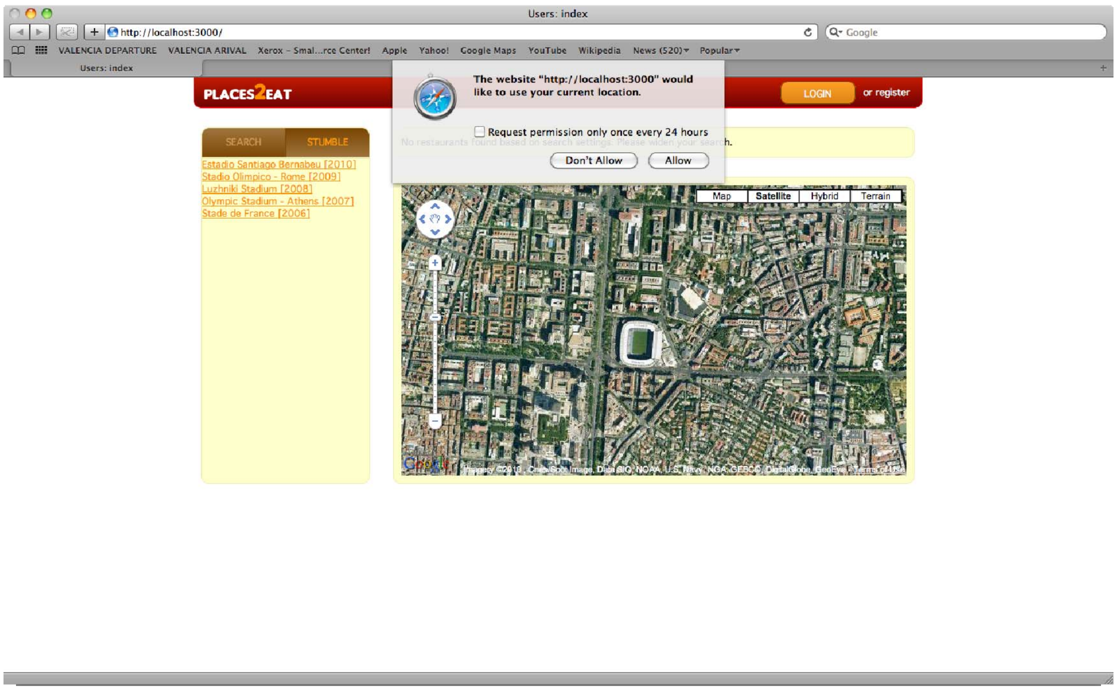
Figure 23 shows the homepage of the application, map switched to road view, no filters used and displaying 5 restaurant markers based on search results according to current map bounds.



localhost wants to know your location. Learn More...

No restaurants found based on search settings. Please widen your search.

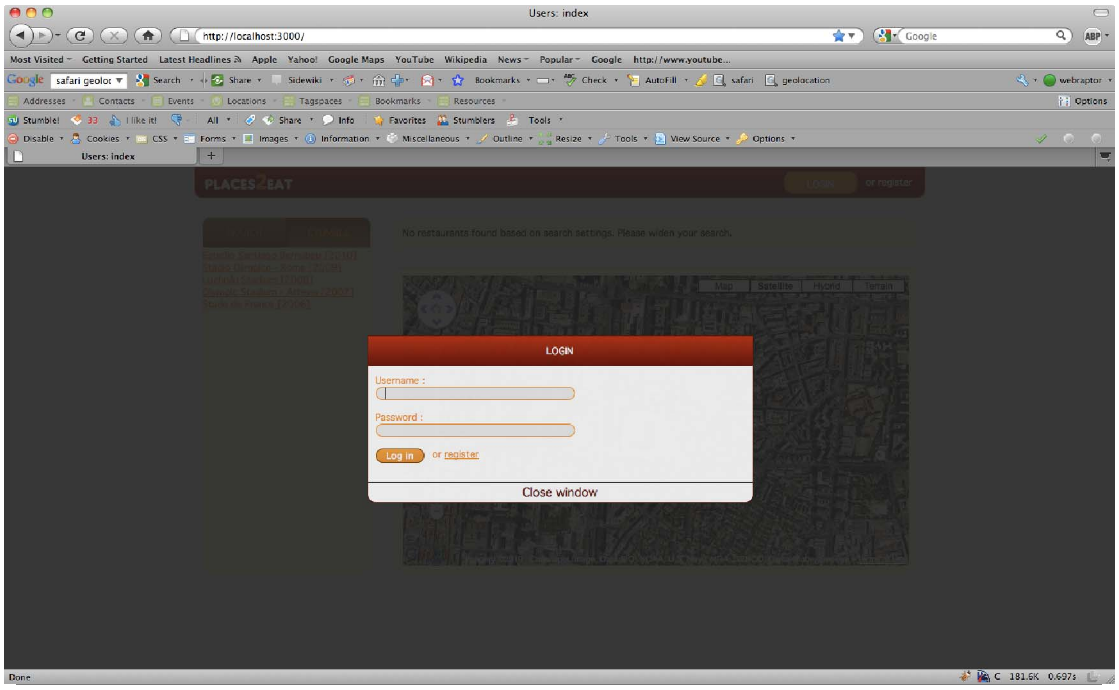
Figure 20. Mozilla Firefox screenshot of Web application’s home page. Browser is prompting user for location sharing as the application has requested it



The website "http://localhost:3000" would like to use your current location.

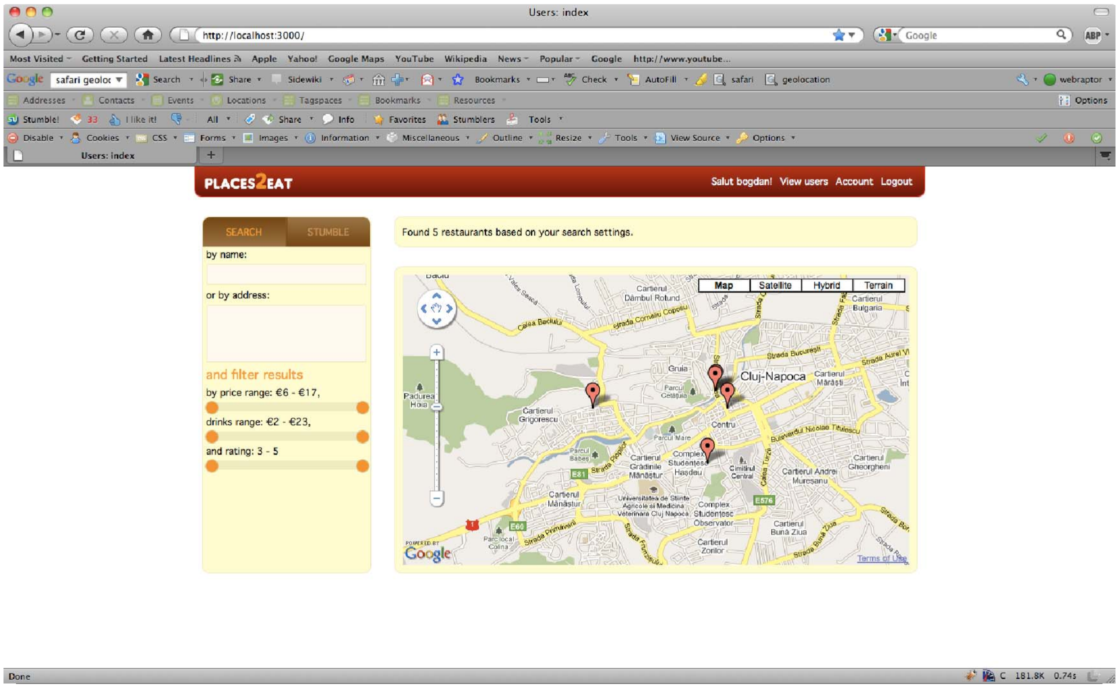
Request permission only once every 24 hours

Figure 21. Safari 5 (latest version release 7th of June 2010) screenshot of the Web application’s homepage, asking permission to share user location with the application



No restaurants found based on search settings. Please widen your search.

Figure 22. Login modal window of Web application



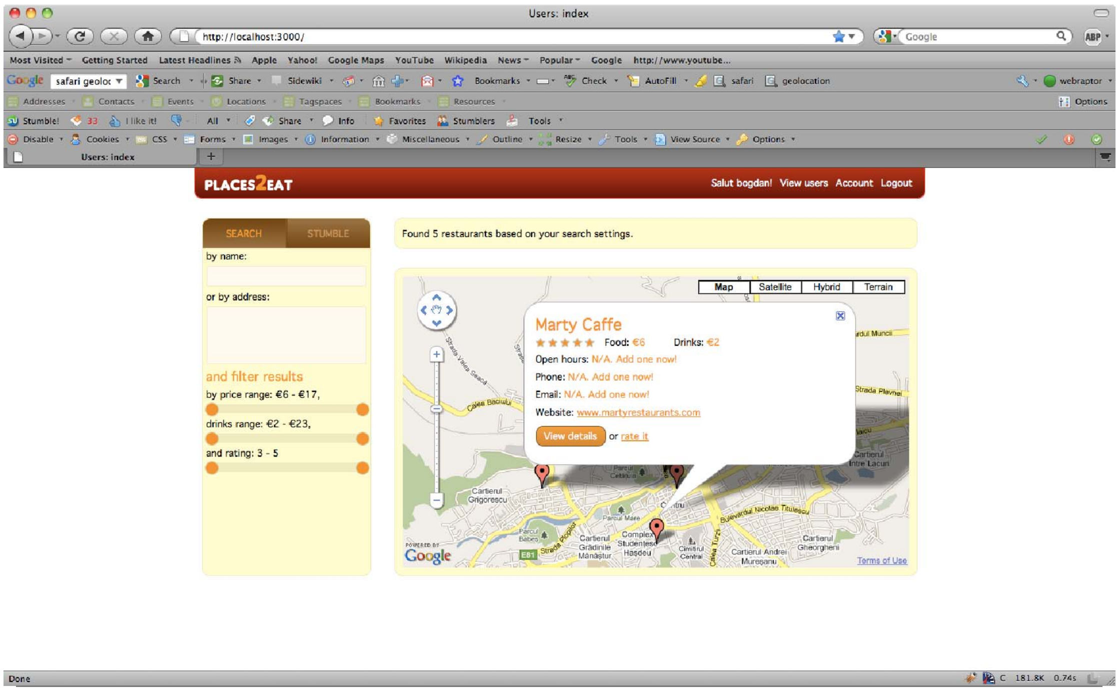
Found 5 restaurants based on your search settings.

Figure 23. Web application, map in road view, with 5 restaurant markers displayed on it.

Figure 24 displays a screenshot of the Web application that is displaying a popup box with details on a restaurant whose marker was previously clicked. The popup displays name, rating, pricing information, as well as open hours, contact phone, email and Website. Additionally, a link to a detailed page about the restaurant is provided.

Figure 25 displays the Web application with filters being used to refine restaurant searches. In order to refine search results, restaurant name was typed, as well as different values selected for pricing and rating.

Finally, Figure 26 displays a user's list view which is available to administrators of the Web application, showing pagination features and hover controls.



Found 5 restaurants based on your search settings.

by name:

or by address:

and filter results

by price range: €6 - €17,

drinks range: €2 - €23,

and rating: 3 - 5

Marty Caffe

★★★★★ Food: €6 Drinks: €2

Open hours: N/A. Add one now!

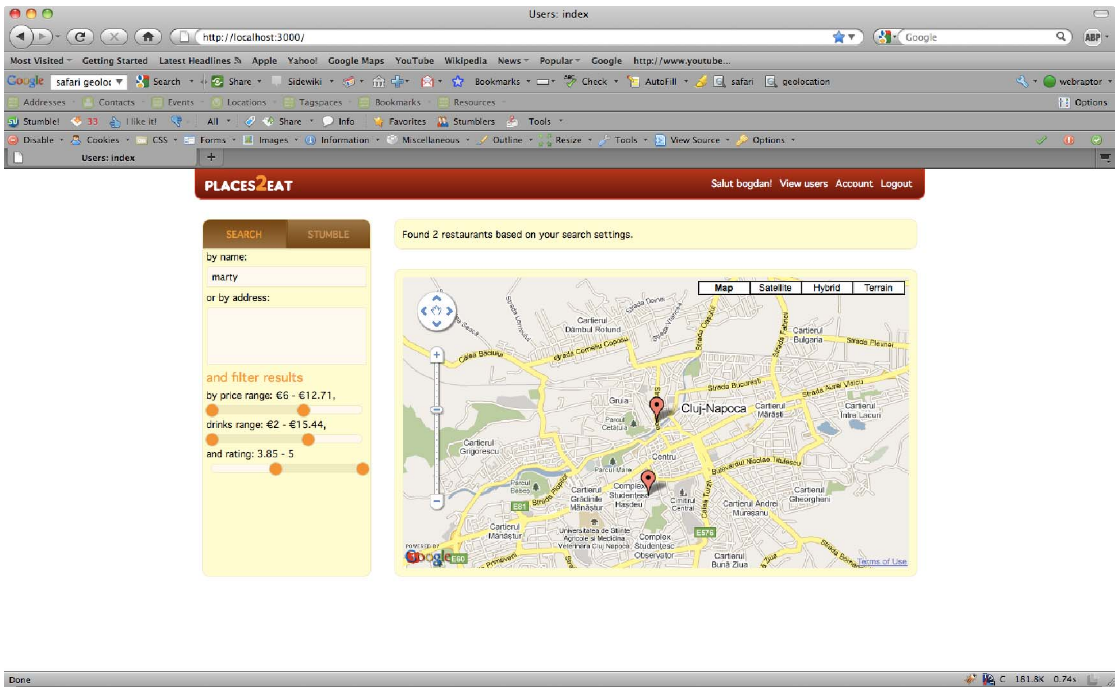
Phone: N/A. Add one now!

Email: N/A. Add one now!

Website: www.martyrestaurants.com

View details or rate it

Figure 24. Popup displaying restaurant information and links to detailed page



Found 2 restaurants based on your search settings.

by name: marty

or by address:

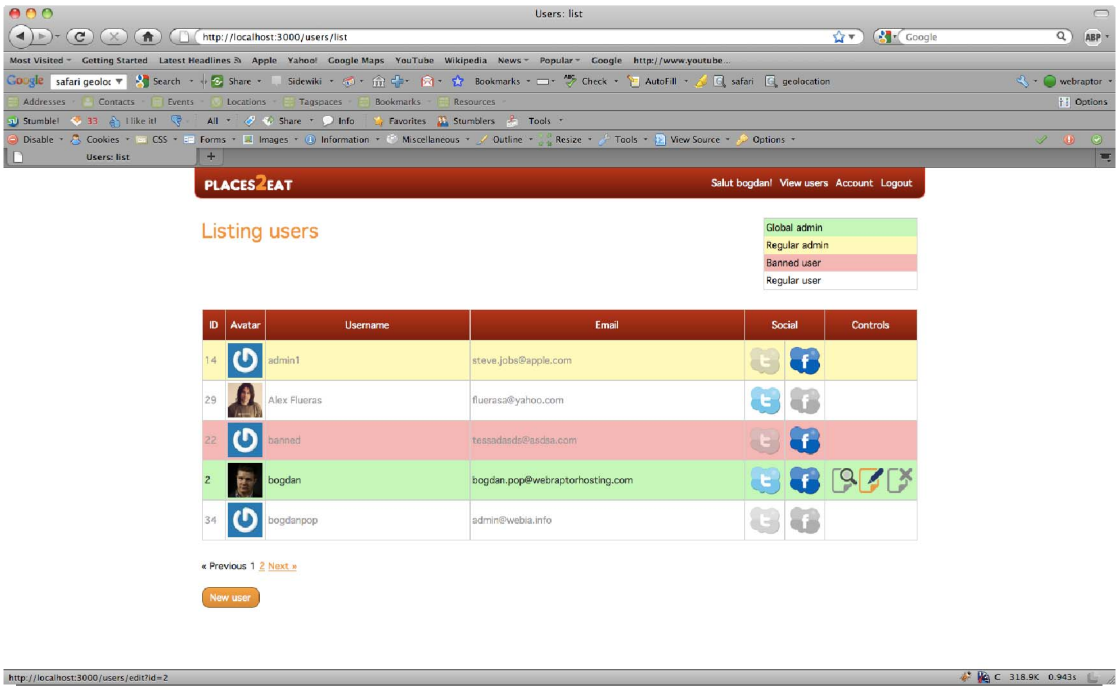
and filter results

by price range: €6 - €12.71,

drinks range: €2 - €15.44,

and rating: 3.85 - 5

Figure 25. Search results refined by using different filters


Users: list
http://localhost:3000/users/list
PLACES2EAT
Salut bogdan! View users Account Logout
http://localhost:3000/users/edit?id=2

Listing users

Global admin
Regular admin
Banned user
Regular user

ID	Avatar	Username	Email	Social		Controls
14		admin1	steve.jobs@apple.com			
29		Alex Flueras	fluerasa@yahoo.com			
22		banned	tessadasds@asdsa.com			
2		bogdan	bogdan.pop@webraptorhosting.com			
34		bogdanpop	admin@webia.info			

« Previous 1 2 Next »

New user

Figure 26. Administrator users list view, with hover controls enabled on 4th user

References

[1] *"Introduction to links and anchors",* HTML 4.01 specification, World Wide Web Consortium (http://www.w3.org/TR/html401/struct/links.html#h-12.1)
[2] Andrew S. Tanenbaum, *"Computer networks", 4th edition*, Prentice Hall, 2003
[3] Dr. Lawrence G. Roberts, *"Resource Sharing Computer Networks",* Advanced Research Projects Agency, Office of the Secretary of Defense, June, 1968
[4] Michael Hauben, *"History of ARPANET",* Columbia University, 1996
[5] Tim Berners-Lee, *"Weaving the Web",* Harper Business, 2000
[6] *"Inventor of the Week Archive: The World Wide Web",* Massachusetts Institute of Technology, MIT School of Engineering, 1999
[7] http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html
[8] *"Ten Years Public Domain for the Original Web Software",* http://tenyears-www.Web.cern.ch/tenyears-www/Welcome.html
[9] http://www.w3.org/LineMode/
[10] John Schneidawind, *"Big Blue unveiling",* USA Today, 1992
[11] Steve Jobs speech during MacWorld keynote address, June, 2007
[12] *"Market Share: Smartphones, Worldwide, 1Q08",* Gartner, June 2008
[13] *"Dataquest Insight: Market Share for Mobile Devices, 1Q09*", Gartner, May 2009
[14] *"Feature Phone",* Phone Scoop, May, 2010
[15] *"Introduction to Future of Web Apps, 2010",* Future of Web Design Conference, May 17-19, 2010
[16] Mike Maznick, *"Nanotech Excitement Boosts Wrong Stock",* The Market, 2003
[17] Tim O'Reilly, *"What Is Web 2.0",* O'Reilly Media, 2005
[18] J. Governor, D. Hinchcliffe, D. Nickull, *"Web 2.0 architectures"*, O'Reilly Media, Sebastopol, May, 2009
[19] Amit Agarwal, *"Web 3.0 concepts explained in plain English",* labnol.org, 2009
[20] Andrew Keen, *"The Cult of the Amateur",* DoubleDay, 2007
[21] Jessi Hempel, *"Web 2.0 is so over. Welcome to Web 3.0",* CNN, 2009
[22] Conrad Wolfram, *"Communicating with apps in Web 3.0",* IT PRO, 2010
[23] John Smart, "Metaverse Roadmap: Pathways to the 3D Web", metaverseroadmap.org, 2006
[24] Tim Berners-Lee, *"The year open data went worldwide",* TED University Conference, 2010
[25] Sven Lennartz , Vitaly Friedman, *"The Smashing Book",* Smashing Media GmbH, DE Druck Europa GmbH, Germany, 2009
[26] Eric Meyer, *"Picking a Rendering Mode",* 2002
[27] Luke Wroblewski, *"Web form design – filling in the blanks",* Rosenfeld Media, 2008
[28] Bogdan Pop, *"Forms. Can't live with them. Can't live without them",* Web International Awards, 2009
[29] Adit Gupta, *"Applying Mathematics To Web Design",* Smashing Magazine, 2010
[30] "ECMAScript Language Specification", ECMA International
[31] Rachel Andrew, Kevin Yank, *"Everything You Know About CSS Is Wrong!",* SitePoint Pty. Ltd., 2008
[32] Steve Grosvenor, "The Flash Anthology: Cool Effects & Practical ActionScript", SitePoint Pty. Ltd., 2005
[33] Steve Jobs, *"Thoughts on flash",* Press release of Apple Inc., April 2010
[34] *"Software and System Requirements for JavaFX Technology",* Sun Developer Network, 2010

[35] *"Silverlight for Mobile",* silverlight.net
[36] Adam Trachtenberg, *"Upgrading to PHP 5",* O'Reilly Media, 2004
[37] Cristian Darie, Wyatt Barnett, *"Build Your Own ASP.NET 3.5 Web Site Using C# & VB", 3rd Edition,* SitePoint Pty. Ltd., 2008
[38] Yukihiro Matsumoto, "*Ruby in a Nutshell",* O'Reilly Media, 2001
[39] Adrian Holovaty, Jacob Kaplan-Moss, *"The Definitive Guide to Django: Web Development Done Right",* Apress, 2008
[40] *"CakePHP: The CookBook",* http://book.CakePHP.org/
[41] Edgar F. Codd, *"A Relational Model of Data for Large Shared Data Banks",* IBM Research Laboratory, 1970
[42] Mike DeSanti, Jeff Gomsi, *"A comparison of object and relational database technologies",* Object Magazine, 1994
[43] *"iPhone Application Programming Guide",* Apple developer center, Apple inc., March 2010
[44] Refsnes Data, January 2010
[45] *"iPhone Human Interface Guidelines",* Apple developer center, Apple inc., March 2010
[46] *"Introduction to The Objective-C Programming Language",* Apple developer center, Apple inc., March 2010
[47] *"Cocoa Fundamentals Guide",* Apple developer center, Apple inc., March 2010
[48] Harvey Weinberg, *"Accelerometers—Fantasy & Reality",* Analog Device, N/A
[49] Sándor Kabai, *"Gyroscope*", Wolfram Demonstrations Project, N/A
[50] Jani Jarvinen, Javier DeSalas, Jimmy LaMance, *"Assisted GPS: A Low-Infrastructure Approach",* GPS World, 2002
[51] *"Geolocation API Specification",* World Wide Web Consortium, 2009
[52] *"The Google Maps JavaScript API V3",* Google inc., 2010
[53] Bogdan Pop, *"Developing a Location-Aware Web-Application for Showcasing Near-by Restaurants",* IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2010, May 28-30 2010
[54] Chris Mills, *"Find me! Geolocation-enabled Opera build",* Opera labs press release, 2009
[55] Oliver Bimber, Ramesh Raskar, *"Spatial Augmented Reality: Merging Real and Virtual Worlds",* A K Peters, 2005
[56] R. Azuma, *"A Survey of Augmented Reality",* 1997
[57] Peter Cooper, *"Beginning Ruby: From Novice to Professional",* Apress, 2007
[58] *"Real applications live in the wild*", RubyonRails.org
[59] Dave Thomas, David Heinemeier Hansson, *"Agile Web development with Rails", 2nd edition,* The Pragmatic Programmers, 2007
[60]David Flanagan, Yukihiro Matsumoto, *"The Ruby Programming Language",* O'Reilly Media, 2008
[61] Dave Thomas, *"Programming Ruby", 2nd edition,* The Pragmatic Programmers, 2005
[62] Rob Orsini, *"Rails Cookbook",* O'Reilly Media, 2007
[63] Mike Beedle, Dave Thomas, "Agile Manifesto", 2001
[64] P. Abrahamsson, J. Warsta, *"New Directions on Agile Methods: A Comparative Analysis",* Proceedings of International Conference on Software Engineering, ICSE, 2003
[65] Matt Stephens,Doug Rosenberg, *"Extreme Programming Refactored",* Apress, 2003
[66] Laurie Williams, *"The Costs and Benefits of Pair Programming",* Proceedings of International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000), 2000
[67] Ken Schwaber, *"Agile Project Management with Scrum",* Microsoft Press, 2004
[68] Peter Bell, *"Solo scrums",* Application Generation, 2007

[69] Staffan Nöteberg, *“Pomodoro Technique Illustrated: The Easy Way to Do More in Less Time”,* The Pragmatic Bookshelf, 2009
[70] Myles Eftos, *“RESTful Rails”,* SitePoint Pty. Ltd., 2008
[71] Roy Thomas Fielding, *“Architectural Styles and the Design of Network-based Software* Architectures”, University of California – Irvine (UCI), 2000
[72] Kevin Marshall, *“Web services on Rails”,* O’Reilly Media, 2006
[73] *“PHP manual”,* www.PHP.net
[74] *“About Python”,* www.Python.org/about
[75] Erik Wrenholt, *“Ruby, Io, PHP, Python, Lua, Java, Perl, Applescript, TCL, ELisp, JavaScript, OCaml, Ghostscript, and C Fractal Benchmark”,* Timestretch, 2005
[76] Tim Hentenaar, *“Benchmark: PHP vs. Python vs. Perl vs. Ruby”,* 2008
[77] Armin Kroll, *“JSON and RESTful Web services on the iPhone”,* jTribe Holdings Pty Ltd, Australia, 2008
[78] Alasdair Allan, *“Learning iPhone programming”,* O’Reilly Media, 2010
[79] Bogdan Pop, *“Developing a Location-Aware Web / Mobile System for Showcasing Near-by Restaurants”,* Sesiunea de Comunicări Ştiinţifice ale studenţilor, UBB, UTCN, June 2010
[80] *“Safari Web content guide”,* Apple developer center, Apple inc., June 2010