Web-based Applications: Extending the General Perspective of the Service of Web

Conference Paper · August 2017

CITATIONS
11

READS
11,529

2 authors:

Nalaka Ruwan Dissanayake
Sri Lanka Institute of Information Technology
37 PUBLICATIONS **88** CITATIONS
SEE PROFILE

Kapila Asanga Dias
University of Colombo
91 PUBLICATIONS **224** CITATIONS
SEE PROFILE

Web-based Applications: Extending the General Perspective of the Service of Web

NR Dissanayake[1#] and GKA Dias[1]

[1]*University of Colombo School of Computing, Colombo 7, Sri Lanka*

[#]nalakadmnr@gmail.com

Abstract*—It has been almost three decades since the introduction of the service of web; meantime various related concepts like Service Oriented Architecture, Cloud Computing, and Internet of Things have been amalgamated to the domain of the web based development. Yet, the view of the service of web and the definitions of the basic related concepts have not been adequately updated. The notion behind the generic term "web application" and its definition do not cover the diverse types of applications available these days, which utilize the service of web. This paper provides an overview of the generic web applications, and then extends the discussion through a contemporary perspective, proposing the umbrella term "web-based application" to address a wider range of systems, which are not likely to be covered by the term "web application". Then the paper proposes a definition for it, focusing on the abstract architectural formalism of the applications, which use the service of web. The paper also discusses the development techniques and technologies, aligning to the proposed definition. In future, we expect to further extend this knowledge towards defining the modern face of the web-based applications, which is the Rich Internet Application.

***Keywords*—Browser, Desktop, Techniques and Technologies, Web**

I. Introduction

The concept of the computer networks runs back to early 60s (Leiner, et al., n.d.), and since then different types of services for network based systems like file sharing and email had being introduced enabling a new breed of computer based systems called distributed systems. Compared to the standalone computer applications – which entirely run in a single computer – these distributed systems contain multiple components running in different computers, which communicate over networks, based on the client-server architecture. In this setting, the concept of the service of web was introduced in early 90s (CERN, 1993), and it became popular, allowing the features of other types of network based services to be delivered via the service of web. Nowadays the client-server based distributed systems commonly known as "web applications" are mainly built exploiting the service of the web, and this domain is evolving and expanding rapidly.

However, during our ongoing research we noted that the knowledge related to the domain of the web is disorganized and it's not easy to understand the relationships between the diverse concepts within the domain. Moreover, we identified that the perspective of the concept of web application has been evolved over time, yet the definitions have not been adequately updated. We noted that the notion behind the generic term "web application" and its definition do not cover the diverse types of applications available these days, which utilize the service of web.

The definitions of the concepts are important as they provide precise common understanding of the focused subject, which helps in proper utilization of the concept. In this paper we propose the umbrella term "web-based application" to address a wider range of systems, which are not likely to be covered by the general term "web applications" and its definition. Then we propose a definition for the term "web-based application", focusing on the common abstract architectural formalism of these system, which are based on the service of web. Additionally, in the direction of supporting the development, we discuss the development TTs of the web-based applications, aligning to the proposed definition.

We conducted a literature survey focusing on the basic concepts of the web, like protocols and architectures, to gain deep understanding of the contemporary application of these concepts. In parallel, we conducted a series of experiments to gain empirical evidence towards understanding the use of these concepts into the development. The experiments were prototype based and we utilized HTML, JavaScript, jQuery, C#.Net, and Android for client-components development, and PHP, JAVA. C#.Net for server-components development.

Section II of the paper provides a background to the rest of the paper, discussing the overview of the web applications and the evolution of them. Section III extends the discussion proposing and defining the concept of web-based application, in a contemporary perspective; and the section also discusses the types of the web-based applications and limitations of them. Section IV discusses the development techniques and technologies (TTs) for different components of the web-based applications, aligning to the proposed definition.

Section V concludes the discussion, also specifying the future work.

II. Background

This section provides an overview of the web applications, distinguishing them from desktop applications. Then the evolution of the web applications is discussed, highlighting how they have grown beyond the notion of the term "web applications".

A. Desktop Applications vs. Web Applications

A desktop application is a program that runs standalone in a computer device, like a desktop computer, laptop, or even a mobile device (PCMag, n.d.). All the components of these applications are located within the device, and these components are executed within a single address space of the system memory.

The components of the web applications are distributed; hence the system is executed in multiple address spaces. PCMag's definition for the web application says that a web application is "*an application in which all or some parts of the software are downloaded from the Web each time it is run. It may refer to browser-based apps that run within the user's Web browser, or to 'rich client' desktop apps that do not use a browser, or to mobile apps that access the Web for additional information*" (PCMag, n.d.). We think that the "rich client applications" or the "Rich Internet Applications" have their own domain, step above the domain of the web applications. Therefore, according to the definition, we can interpret that the scope of the web applications is limited to the browser-based applications.

B. Evolution of Web Applications

Early systems, which utilized the service of web were called "Web sites", and they were limited to a collection of documents with static content, interconnected via hyperlinks (Jazayeri, 2007). These documents were located in a web server, and the user can request for these documents using a web client application, mainly the web browser. Upon users' needs the web browser sends requests to the server, and displays the received responses to the user.

Later these web sites incorporated server-side development languages like PHP or JAVA, and related TTs; as well as client-side processing languages such as JavaScript (JS), and associated TTs. With all these TTs web sites were evolved to web applications, with server-side and/or client-side application components, which are capable of processing and producing dynamically tailored information (Shklar & Rosen, 2003). With continuous evolvements, the web applications nowadays provide a platform to deliver features of some other services – which can be gained from data networks, such as e-mails, and file transferring – via web protocols. With all these features, the service of web and the web applications have become advanced, powerful, and a complex set of entities, with a high demand.

A web application can serve multi-users in parallel, with less hassle compared to the desktop applications. As Conallen (Conallen, 1999) says, "*One of the most significant advantages of a web application is its deployment*". He continues saying that "*deploying a web application is usually a matter of setting up the server-side components on a network. No special software or configuration is required on the part of the client*". In such setting, web applications are easy to manage, maintain, and modify, with a low workload compared to the higher number of users it can serve simultaneously. In clients' perspective, no special software or configuration is required; and for the users, the web applications provide both platform and location independent access. In early stages, there were some device dependencies for the web applications regarding the mobile devices; however, the modern TTs are minimizing these barriers and evolving rapidly.

New types of system like mobile applications (Techopedia, n.d.), service oriented systems (Bianco, et al., 2011), and Internet of Things (Agenda, 2016) have become popular, which use the service of web for communication. We noted that these types of applications are not covered by the definition of the "web application" and they can be seen as much complex systems than the standard traditional web applications. However, they all share similar architectural characteristics within the domain of web.

There are several architectural styles available for the systems, which utilize the service of web: client-server or two-tier architecture, three-tier architecture, multi-tier architecture, and Service Oriented Architecture (SOA) are some regularly used styles. All these styles are based on the basic client-server model and then are extended up to complex multi-tier or SOA models, to serve the needs of the advanced user requirements. The main characteristic of these styles is, they partition the system into different layers, where these layers contains different components. These different components in the system, run in multiple locations and environments (mainly – and at least – in the client and the server), and the development of these components are heavily TTs dependent. This setting initiates generating the complexity in web applications; these facts – location and technology dependencies in development, complex structures, variable size, and advanced features and capabilities – together have made the systems, which use the service of web, a very advanced and complex.

III. The concept of Web-Based Applications

It was discussed in the section II.A, the term "web application" covers the basic notion of browser based client-server applications. As discussed in the section II.B, in present, the use of the service of web has

expanded its limits, beyond the generic web applications, and beyond the web browser. Therefore, we think that the term "web application" or its definition are not capable of addressing all these systems within a wider range.

However, as discussed in the later part of the section II.B, still the architecture of these systems are based on the client-server style, using the request-respond model over HTTP; thus, they can be grouped into a single domain. We propose the term "Web-based Applications" to refer all these applications, which are based on the client-server model and utilize the service of web for communication. This term can be seen as an umbrella term, which covers a wider range of systems than the term "web application", both in and beyond the browser, addressing a superset of standard web applications.

During the literature survey, we identified an available definition for the term "web-based applications". The Techopedia (inc., 2017) defines the web-based applications as: "*A web-based application is any program that is accessed over a network connection using HTTP, rather than existing within a device's memory. Web-based applications often run inside a web browser. However, web-based applications also may be client-based, where a small part of the program is downloaded to a user's desktop, but processing is done over the internet on an external server. Web-based applications are also known as web apps*." They further explain that "*there is a lot of confusion created by the use of terms like web-based, internet-based and cloud-based when referring to applications. Web-based applications actually encompass all the applications that communicate with the user via HTTP. This includes light applications like Flash games, online calculators, calendars, and so on, as well as more intensive applications such as word processors and spreadsheet applications*" (inc., 2017). This definition and the explanation widen the general notion of the web applications, and we think that this definition can be improved by constructing it in an architectural point of view, allowing it to have more space to be extended by merging variety of related concepts into the domain.

A. A Contemporary Definition for the Web-based Applications

Before defining the web-based applications, we had better understand the formalism of them. As stated in section II.A, a desktop application is an application that runs standalone in a desktop/laptop computer, mobile device (PCMag, n.d.), or even in any other device as an embedded application. Remote components can be added to the standalone applications, utilizing web technologies for communication, data processing, storing, etc. Such systems, which utilizes the service of the web, can be considered as the "Web-based Applications". Note that the web browser itself is a desktop application, which utilizes the service of web.

Based on this discussion, we propose to define the web-based applications as follows. This definition also specifies the scope of the capabilities of web-based applications.

> Web-based application is a system, with application component(s) in client-side [client-component(s)], which communicate(s) with application component(s) in a web server [server-component(s)], for processing data. They utilize the service of web, based on the client-server architecture, request-response model, standard HTTP, and other related techniques and technologies.

Elaborating this definition further, we suggest to consider this basic model of a web-based application as a "Standalone Web-based Application". This concept is presented by the formula below.

Standalone desktop application (Client components) + Web server layer (Server components)	=	Standalone web-based application

This standalone web-based application can be extended by adding more layers/partitions/tiers – mainly to the server-side – and the system can be developed into a multi-tier web-based application. This concept is specified in the formula below.

Standalone web-based application + External layers	=	Multi-tier web-based application

The need for introducing this concept is to cater the possibilities of extending the technological and technical capabilities of the web-based applications.

B. Types of Web-based Applications

This section discusses the types of web-based applications towards providing a good understanding of the scope of the proposed term "web-based applications" and its definition.

Taivalsaari and Mikkonen had also come up with a similar idea, and they have introduced a taxonomy for the concept they call "Web-based Software" (Taivalsaari & Mikkonen, 2011). In their taxonomy, they have used the dimension in three categories: "*those that are related to the characteristics of the (1) applications, (2) execution environment, and (3) end to end architecture*." The key focal point of their discussions is Cloud Computing, and they have discussed very little beyond the standard browser-based web applications.

Puder (Puder, 2004) had introduced a framework in similar context. His attempt is to migrate arbitrary desktop applications to web platform, without requiring any changes in the source code. As he says, "*it also allows the development of new web applications based on well understood paradigms, thereby making it unnecessary to understand web technologies*." Instead of improving the knowledge of the web, his approach takes the traditional web engineering to a new level. However, the applicability of this approach is questionable.

We would like to categorize the web-based applications, based on the types of the client-components. The client-components can be either browser-based or non-browser-based; therefore, we classify the web-based applications into two groups: browser-based applications and non-browser-based applications.

Browser-based applications are commonly available in WWW; some good examples can be given as Google applications and Facebook. The client-components of the browser-based applications are executed in a web browser. These client-components are downloaded by the browser, at the time the user starts using the application and continue executing till the user closes the browser.

The non-browser-based applications are not commonly available. Some desktop applications such as download managers like DAM (Corporation, 2016), do support HTTP, therefore can be considered as non-browser-based applications. The non-browser-based applications can be useful in special scenarios. Note that the web browser itself is a desktop application, which communicates with web servers via HTTP. Similar to the way the web browsers work, a desktop application can be exploited as a client-component to access the web servers and utilize the service of web. The advantage here over the browser-based client-components is that a native desktop application can contain rich and complex GUIs, and they can be developed with less effort compared to the browser-based web apps.

This is not a widely used technique for desktop/laptop types of devices, and also have all the disadvantages related to the management and maintenance of the desktop applications compared to browser-based web applications. However, still in some scenarios it can be really helpful and also feasible. This concept is usually used for mobile apps. Additionally, these non-browser-based application can be some embedded application components in some hardware systems, and such systems fall into a modern concept called Internet of Things.

C. *Limitations of Web-based Applications*

Despite the advantages of the web applications over desktop applications, the browser-based web applications have some limitations; and several limitations of them are also valid for all the types of web-based applications. Manu (Manu T S, 2011), Preciado *et al*. (Preciado, et al., 2005), and Mesbah and Deursen (Mesbah & Deursen, 2007) had dicussed these limitation in detail, and we provide an analysis of them in the context of web-based applications, under five main facts.

1. Poor GUIs: This affects the browser-based applications, who are constrained by HTML thus lack in rich GUIs compared to desktop applications (Manu T S, 2011). Due to poor GUIs, mostly the operations of a single feature is scattered across multiple pages, so the user has to navigate through series of pages to complete a single task (Preciado, et al., 2005), experiencing the traditional page sequence paradigm (Mesbah & Deursen, 2007).

2. Slower response: Since the browser-based applications use the work-wait pattern, their interactions are restricted with page refreshes (Manu T S, 2011) (Preciado, et al., 2005). The page sequence and refresh setting increases the time required to complete a task, resulting slow task completion.

3. Lower user experience: The cumulative effect of the above two facts, leads the browser-based applications to deliver poor user experience.

4. Lack of management, maintenance, and modification: The non-browser-based client-components can eliminate the aforementioned facts with their rich GUI development capabilities; however, they do not fit into most scenarios, due to their lack of management, maintenance, and modifications, similar to desktop applications compared to browser-based systems.

5. Communication limitations: The non-browser-based client-components need dedicated web based communication TTs, which limits the autonomy of the diversity of the combinations between the client and different types of server components like web services.

Overcoming these limitations of web-based applications, a new breed of applications named Rich Internet Applications (RIAs) had been introduced, outlining a new era for web, named Web2; and they have become increasingly important and popular. Discussions of the RIAs are intentionally avoided in this paper.

IV. Techniques And Technologies for Web-Based Applications Development

To understand the full potential of the capabilities of systems and address the development complexities, adequate understanding of the development TTs and their usage is essential. Web TTs have been discussed by many, in different perspectives already (Fraternali, 1999) (Jazayeri, 2007) (Doyle & Lopes, 2008). Our

attempt of classifying the TTs is focusing on the architectural formalism of the web-based applications, aligning to the proposed definition, based on Fielding's concept of architectural elements (Fielding, 2000): Components, Connectors, and Data. Having said that, when discussing about the TTs used for the development of the web-based applications, considering only the client-side or server-side components may not provide complete realization; in addition, we had better look into all the aspects related to the web-based systems, including the Connector and Data elements.

A. TTs for the Client-Components of Web-based Applications

The TTs of the client-components are discussed under the categories: 1) browser-based, and 2) non-browser-based.

1) TTs for the Browser-based Client-Components

Web page is the basic and the main resource of a web site or a browser-based web application, which is loaded into the browser in the client-side, providing an interface to the user to interacts with the system through a GUI. Mostly in web applications, both the client-components and the server-components are developed into the same web page. If the business logic is entirely separated from the web page, still some components are needed to be developed into the web page in order to utilize the components with business logic. These components developed in the web page are executed in the server, before the web page is sent to the clients' browser, and they are developed using the sever-side TTs, discussed in the section IV.B.

The web browser – which is a desktop application – is utilized by the user to communicate with the server, access and fetch the web pages, and display the content of the web page. The web browser provides a platform for the execution of the client-side components of generic browser-based applications. Even though the web browser itself is a platform dependent desktop application, like any other desktop application, the platform it provides for the client-components of the browser-based applications, helps the web-based applications to be platform independent. In this setting, the browser works as a middleware for the browser-based applications. Detailed discussions about the web browsers are intentionally avoided, since the actual focus is on the application components, which run on the browser.

The standard web TTs used for the development of the client-components are: HTML to develop the content including GUIs, CSS to format the content towards pleasant presentation, and JS to develop the behaviour for processing and manipulating the content. Additionally, related frameworks and libraries are used to develop enhanced features. Mostly these frameworks/libraries are JS based, and they generate and manipulate HTML and CSS. jQuery and Bootstrap are some popular libraries for browser-based client-components development.

With the evolution of JS, the browser-based web application have become very advanced systems (Jazayeri, 2007). However, they are still not without limitations. One face of these limitations is related to the GUIs and the presentation of the content. Developing desktop applications like rich GUIs is not easy with browser-based applications. In early stages, plug-in based techniques like JAVA applets and Macromedia (later Adobe) Flash became popular with their rich GUI development features. These proprietary technologies were with their own layer of technical issues and later became less demanded as standard HTML, CSS, and JS based TTs evolved. Yet the complexity, difficulty, and the workload of developing rich GUIs for browser-based applications is relatively higher than the desktop applications.

Additionally, the browser-based client-components can be developed as a complete application, without server-side components; therefore, can be utilized in a way of the standalone desktop applications. They can be placed within the file system of a client computer without hosting in a web server, and run them in the web browser without needing a network. This can be seen as another way of developing standalone desktop applications using browser-based TTs, introducing a new classification for desktop applications, as browser-based desktop applications and non-browser-based desktop applications. Currently this type of browser-based desktop applications are utilized for systems like documentation readers for software packages, which are contained in portable disks like DVDs or CDs. This concept is out of the scope of this paper, thus will not be discussed further.

2) TTs for the Non-Browser-based Client-Components

Focusing on the desktop or laptop computers, one approach for the development of the non-browser-based client-components is using a dedicated technology, enriched with native web utilities, like Adobe Flash/Flex or JAVA applets. Other approach is using the standard native desktop application development frameworks like .Net or JAVA; they also contain utilities for developing web enabled features, which are capable of communicating with web servers.

In mobile application development, the approach called "Hybrid Applications" (Bristowe, 2015) are becoming popular. These hybrid applications either use a browser container inside the native application, to utilize a browser-based application; or the native application itself communicates directly with the web servers. However, a native desktop application with a browser container will act almost like another browser, hence

does not meet the focus of the context of this discussion. Most mobile apps nowadays can be seen as client-components of web-based applications, since they utilize network and the service of web to implement advanced functionalities, such as location based services, emails, etc.

A contemporary concept called Internet of Things has become popular, which enable network utilization and data communication for hardware components such as watches, vehicles. etc. (Wikipedia, n.d.). Various sensors in these hardware systems read parameters and the embedded client components in these systems communicate with server components for processing these parameters and to provide many advanced features. Deep discussions of the IoT concept is avoided here, focusing only on the client component development TTs. There are dedicated TTs available for client component development of IoT systems, such like frameworks, platforms, standards, etc. (Wikipedia, n.d.) (Maniappan, 2015).

B. TTs for the Server-Components of Web-based Applications

A web server is a process running in a dedicated remote computer (Dye, et al., 2008), which provides a platform to host web-based systems and execute the server-components. The web server itself is not considered as a component of a web-based application.

Server-component of a web-based system can be an application program running in a web server, written using server-side language and/or framework, and related TTs. Usually the server-side languages and related TTs are selected based on the type of the web server, or vice versa. JSP based on JAVA, PHP, ASP based on .Net, and python are the major development environments for server-components. PHP is usually used with Apache Web servers, which can be installed in both Microsoft (MS) Windows and Linux operating systems, but mostly used in Linux systems. JSP/JAVA is usually used with Tomcat or compatible web server, again mostly in Linux systems, but also can be used in MS-Windows. ASP/.Net is used in IIS server in MS Windows systems. Likewise, the server-components development TTs are highly coupled with the web server and the operating system of the host where the web server is installed.

The server-components can also be in form of a "Web service", which is a special type of server application, which can be developed using server-side languages and additional dedicated TTs like SOAP, REST, etc. The web services are developed and executed as a standalone application component in a web server, and they provide an Application Program Interface (API) for client-components to communicate with them over dedicated TTs based on HTTP. The domain of the concept of web services is self-contained and large, and in depth details are not covered here.

Service Oriented Architecture (SOA) and Cloud Computing are some advanced concepts related to the web services, and they enable extending the capabilities of the server components of web-based applications. These concepts extend the basic client-server architecture into n-tier architecture.

Regarding the server-side TTs, it should be understood that they are independent from the client-side TTs. However, to be noted, web services can provide a better compatible service for the non-browser-based client-components than standard web server application components.

C. TTs for the Connector Elements of Web-based Applications

Instead of looking into the internal specifications of the communication TTs of the web-based applications, the basic features of them are discussed here. There are several aspects of the connectors to be considered, related to the communication in web-based applications, which are discussed below. For the connector elements used in communication: the client-side contains the connector-components for request sending and response receiving purposes; and the server-side contains the request accepting and response sending components.

1) TTs for Client-side Connector Components of Web-based Applications

For the browser-based client-components, the first request to the server is initiated by the browser itself, to load the initial client-components of the system – usually a web page – to the browser. The features in the loaded client-components are capable of initiating further communication with web server, based on user interactions; and these features use mechanism like HTML hyperlinks, JS redirections, or form submissions to develop the request initiations.

For non-browser client-components, the request sending and response receiving over dedicated TTs like SOAP, XML, or JSON can be developed using the available web utilities in native languages.

2) TTs for Server-side Connector Elements of Web-based Applications

In server, the connector components are expected to receive the request, and send the response – containing the proper resources – back to the client. These features look like the operations of the server itself, but actually what we focus here is on the layer above the web server.

When the web server receives a request, it looks for the targeted resource by the request within the application, and then the execution will be passed to that proper

resource, which can be a web page, a script file, or any other component. The targeted resource contains the server-side connector component(s), which will process the request and may be forward the data to a processing server-component. Once the result is ready and received by the server-side connector from the processing server-components, it will do the needful to prepare the response and release it to the web server, to be sent to the client-side connector component. These connector components are usually developed using the same TTs used to develop the server-components.

D. TTs for the Communication Methods of Web-based Applications

In data communication of the web-based applications, request methods specified in HTTP like GET, POST, etc. are used to send status/data to the server along with the request (Shklar & Rosen, 2003). When the GET method is used, data can be appended to the URL as a query string (Shklar & Rosen, 2003). When the data set is larger and/or contains sensitive data, the POST method can be used; then the data will be included into the body of the request (Shklar & Rosen, 2003). Selecting the correct request method according to the HTTP specification, to satisfy the actual requirements is essential, to increase the semantic of the connector, and also the efficiency of the communication, by minimizing unwanted processing.

As it had been already mentioned, for the communication between the client and the server, the request-respond model is used, and the communication is mainly governed by the HTTP. However, additionally some other protocols like SOAP or techniques like REST can also be incorporated. REST (Fielding, 2000) style uses a way of constructing the URL to specify some state related data, to be sent to the server. A REST enabled server-side components of the traditional web application can be utilized in a form of a web service, without any other specific protocols like SOAP, which can be seen as a great advantage of the REST.

E. TTs for the Data elements of Web-based Applications

In the case of browser-based application, mostly the response is a web page containing the results for the request, or some other file like an image or text file, which the browser will load and display to the user. For non-browser-based application, the response could be a set of data, which the client component(s) can process and display the result to the user.

Data can be formatted for communication, to have a better structure, hence better meaning, which increases the efficiency of the communication and processing. The default data format used in the request methods are based on plain text, and usually the server-side development environments/frameworks contain mechanisms to read data easily. Other than plain text, techniques like XML (Bray, et al., 2006), JSON (T. Bray, 2014), or even HTML can be used to format data.

V. Conclusion

The applications, which utilize the service of web have been evolved into much complex systems, which are not likely to be covered by the term “web application”, and the definition for the “web application” has not been updated to address these modern and complex systems. This paper proposes the umbrella term “web-based application” to cover the systems, which utilize the service of web; and the paper also provides a definition for it, focusing on the abstract architectural formalism of these applications/systems.

This paper has discussed the limitations of the web-based applications, and also the development TTs available for the web-based applications, in a contemporary view, aligning to the abstract architectural components of these applications. This discussion has covered some modern concepts like cloud computing and internet of things under the introduced concept, web-based applications.

In present the development of the standard web-based applications is becoming less as the more powerful RIAs have become popular. We expect to further extend the notion of the web-based applications to accommodate the RIAs and related concepts, and also their development TTs, towards realizing the abstract architectural formalism of the RIAs.

References

Agenda, I., 2016. *Internet of Things (IoT).* [Online] Available at: http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT
[Accessed 16 05 2017].

Bianco, P., Lewis, G., Merson, P. & Simanta, S., 2011. *Architecting Service-Oriented Systems,* s.l.: s.n.

Bray, T. et al., 2006. *Extensible Markup Language (XML),* s.l.: W3C.

Bristowe, J., 2015. *What is a Hybrid Mobile App?.* [Online] Available at: http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/
[Accessed 12 11 2015].

CERN, 1993. *Ten Years Public Domain for the Original Web Software.* [Online] Available at: http://tenyearswww.web.cern.ch/tenyearswww/Declaration/Page1.html
[Accessed 25 10 2015].

Conallen, J., 1999. *Modeling Web Application Architectures with UML,* s.l.: Rational Software.

Corporation, T., 2016. *Download Accelerator Manager.* [Online]

Available at: http://www.damdownloader.com/ [Accessed 20 11 2015].

Doyle, B. & Lopes, C. V., 2008. *Survey of Technologies for Web Application Development.* [Online] Available at: arXiv:0801.2618v1 [Accessed 21 08 2015].

Dye, M. A., McDonald, R. & Rufi, A. W., 2008. *Network Fundamentals.* s.l.:Dorling Kindersly.

Fielding, R. T., 2000. *Architectural Styles and the Design of Network-based Software Architectures,* Irvine: University of California.

Fraternali, P., 1999. Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. *ACM Computing Surveys,* 31(3), pp. 227-263.

inc., t., 2017. *Web-Based Application.* [Online] Available at: https://www.techopedia.com/definition/26002/web-based-application [Accessed 10 02 2017].

Jackson, J. C., 2011. *Web Technologies.* s.l.:Dorling Kindersley.

Jacyntho, M. D., Schwabe, D. & Rossi, G., 2002. A SOFTWARE ARCHITECTURE FOR STRUCTURING COMPLEX WEB APPLICATIONS. *Journal of Web Engineering,* 1(1).

Jazayeri, M., 2007. *Some Trends in Web Application Development.* Minneapolis, s.n.

Lawton, G., 2008. New Ways to Build Rich Internet Applications. *Computer,* August, 41(8), pp. 10 - 12.

Leiner, B. M. et al., n.d. *Brief History of the Internet,* s.l.: Internet Society.

Maniappan, 2015. *List of 10 IoT Frameworks And Platforms.* [Online] Available at: http://www.iotleague.com/list-of-10-iot-frameworks-and-platforms/ [Accessed 05 10 2016].

Manu T S, 2011. *Advances & Applications in Ajax,* s.l.: CUSAT.

Mesbah, A. & Deursen, A. v., 2007. *An Architectural Style for AJAX.* Mumbai, s.n.

PCMag, n.d. *Encyclopedia - Definition of: desktop application.* [Online] Available at: http://www.pcmag.com/encyclopedia/term/41158/desktopapplication [Accessed 30 10 2015].

PCMag, n.d. *Encyclopedia - Definition of: Web application.* [Online] Available at: http://www.pcmag.com/encyclopedia/term/54272/webapplication [Accessed 30 10 2015].

PCMag, n.d. *Encyclopedia - Definition of:application program.* [Online] Available at: http://www.pcmag.com/encyclopedia/term/37919/applicationprogram [Accessed 30 10 2015].

Perrey, R. & Lycett, M., 2003. *Service-Oriented Architecture.* s.l., IEEE.

Preciado, J., Linaje, M., Sanchez, F. & Comai, S., 2005. *Necessity of methodologie to model Rich Internet Applications.* s.l., s.n.

Puder, A., 2004. *Extending desktop applications to the web.* Dublin, s.n.

Shklar, L. & Rosen, R., 2003. *Web Application Architecture Principles, protocols and practices.* England: John Wiley & Sons Ltd.

Stair, R. M. & Reynolds, G. W., 2003. *Principles of Information Systems.* s.l.:s.n.

T. Bray, E., 2014. *The JavaScript Object Notation (JSON) Data Interchange Format,* s.l.: Internet Engineering Task Force (IETF).

Taivalsaari, A. & Mikkonen, T., 2011. *Objects in the Cloud May Be Closer Than They Appear Towards a Taxonomy of Web-Based Software.* Williamsburg, s.n.

Techopedia, n.d. *Mobile Application (Mobile App).* [Online] Available at: https://www.techopedia.com/definition/2953/mobile-application-mobile-app [Accessed 17 05 2017].

Toffetti, G., Comai, S., Preciado, J. C. & Linaje, M., 2011. State-of-the Art and trends in the Systematic Development of Rich Internet Applications. *Journal of Web Engineering,* 10(1), pp. 70-86.

W3C, 2004. *Architecture of the World Wide Web, Volume One.* [Online] Available at: http://www.w3.org/TR/webarch/ [Accessed 28 10 2015].

W3C, n.d. *Help and FAQ.* [Online] Available at: http://www.w3.org/Help/#webinternet [Accessed 25 10 2015].

Ward, M., 2006. *How the web went world wide.* [Online] Available at: http://news.bbc.co.uk/2/hi/science/nature/5242252.stm [Accessed 30 10 2015].

Wikipedia, n.d. *Internet of things.* [Online] Available at: https://en.wikipedia.org/wiki/Internet_of_things#Frameworks [Accessed 10 02 2017].