UDC 519.85:004.42

THE DIFFERENCE BETWEEN DEVELOPING SINGLE PAGE APPLICATION AND TRADITIONAL WEB APPLICATION BASED ON MECHATRONICS ROBOT LABORATORY ONAFT APPLICATION

V. Solovei[1], O. Olshevska[2], Y. Bortsova [3]

[1,2,3]Odessa National Academy of Food Technologies, Odessa, Ukraine
ORCID: [2]0000-0002-4512-3915, [3] 0000-0001-6712-8357
Scopus ID: [2]57192687506,
E-mail:[1] solovei.vitaly@gmail.com, [2]olshevska.olga@gmail.com, [3] bortsova.07@gmail.com

Abstract: Today most of desktop and mobile applications have analogues in the form of web-based applications. With evolution of development technologies and web technologies web application increased in functionality to desktop applications. The Web application consists of two parts of the client part and the server part. The client part is responsible for providing the user with visual information through the browser. The server part is responsible for processing and storing data.

MPA appeared simultaneously with the Internet. Multiple-page applications work in a "traditional" way. Every change eg. display the data or submit data back to the server. With the advent of AJAX, MPA learned to load not the whole page, but only a part of it, which eventually led to the appearance of the SPA. SPA is the principle of development when only one page is transferred to the client part, and the content is downloaded only to a certain part of the page, without rebooting it, which allows to speed up the application and simplify the user experience of using the application to the level of desktop applications.

Based on the SPA, the Mechatronics Robot Laboratory ONAFT application was designed to automate the management process. The application implements the client-server architecture. The server part consists of a RESTful API, which allows you to get unified access to the application functionality, and a database for storing information. Since the client part is a spa, this allows you to reduce the load on the connection to the server and improve the user experience

Анотація: Сьогодні більшість настільних і мобільних додатків мають аналоги у вигляді веб-додатків. З розвитком технологій розробки і веб-технологій веб-додаток збільшилася в функціональності для настільних додатків. Веб-додаток складається з двох частин клієнтської і серверної частини. Клієнтська частина відповідає за надання користувачеві візуальної інформації через браузер. Сервер відповідає за обробку та зберігання даних.

Багатосторінковий додаток з'явилося одночасно з Інтернетом. Багатосторінкові програми працюють «традиційним» способом. Кожна зміна, наприклад, відображення даних або відправлення даних назад на сервер. З появою AJAX багатосторінкові додатки навчилися завантажувати не всю сторінку, а лише частину її, що в кінцевому підсумку призвело до появи односторінкових додатків. Односторінкові додатки - це принцип розробки, коли на клієнтську частину передається тільки одна сторінка, а контент завантажується тільки в певну частину сторінки без перезавантаження її, що дозволяє прискорити роботу програми і спростити використання користувачем додатку до рівня настільних додатків.

Засноване на односторінковому додатку, додаток Лабораторії робототехніки і мехатроніки ОНАХТ було розроблено для автоматизації процесу управління. Додаток реалізує архітектуру клієнт-сервер. Сервер складається з RESTful API, який дозволяє отримати уніфікований доступ до функціональності програми і бази даних для зберігання інформації. Оскільки клієнтська частина являє собою односторінковий додаток, це дозволяє знизити навантаження на з'єднання з сервером і поліпшити роботу користувача

Keywords: web, web application, client-server architecture, MPA, SPA.
Ключові слова: веб, веб-додаток, архітектура клієнт-сервер, багатосторінковий додаток, односторінковий додаток

1. Introduction

Web applications are usually separated into layers that are called "tiers", and each layer has its own role.

Desktop applications usually consist of one architectural layer - a client, web applications are often separated into three levels.

A client layer is usually an application interface that is provided to a user with information through a web browser. At this level, only the simplest business logic of the application is usually rendered. The application server is the middle architecture component that connects the client part and the database. On this layer, most of the business logic of the application is implemented. The database server layer, which provides the storage of information needed for the application, is usually implemented using a database management system. The browser (client) part of the application sends a request to the web server, the server according to business logic processes the request and sends data for writing or deleting to the database. If the client sends a request for data, the server part of the application receives data from the database according to the request and sends it to the client part for display to the user. **Fig. 1**

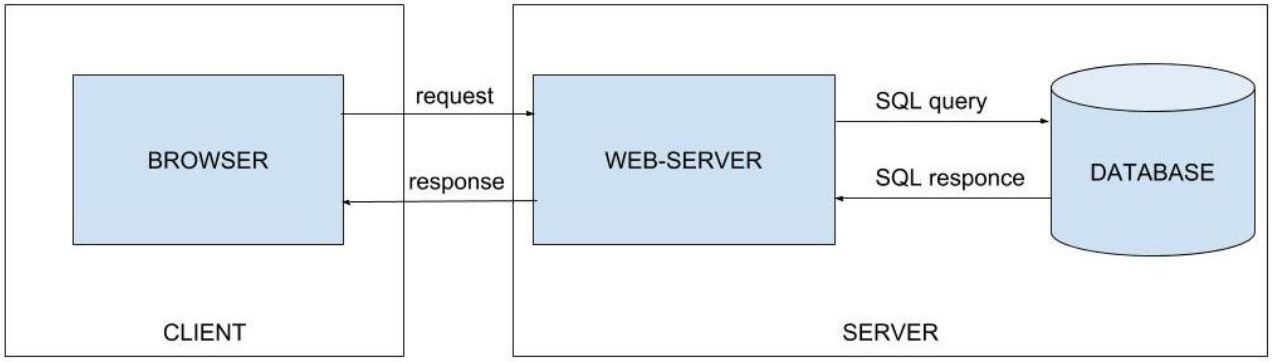

Fig. 1 - the information schema of web-application.

In web development there are two major design patterns that exist for web applications development – one of the oldest and appeared simultaneously with the appearance of the Internet is multi page web applications (MPA) and and modern, which arose with the advent of ajax requests in traditional applications that is called single page web applications (SPA).

2. MPA (traditional web application)

In the simplest form, a multi-page application consists of several pages with a static information (text, images, etc) and links to the other pages with the same content. During a jump to another page, a browser reloads content of a page completely and downloads all resources again, even the components which are repeated throughout all pages (e.g., header, footer). [2] The main technologies for building multi-page web applications are html, css and javascript. It was the first way of developing websites.

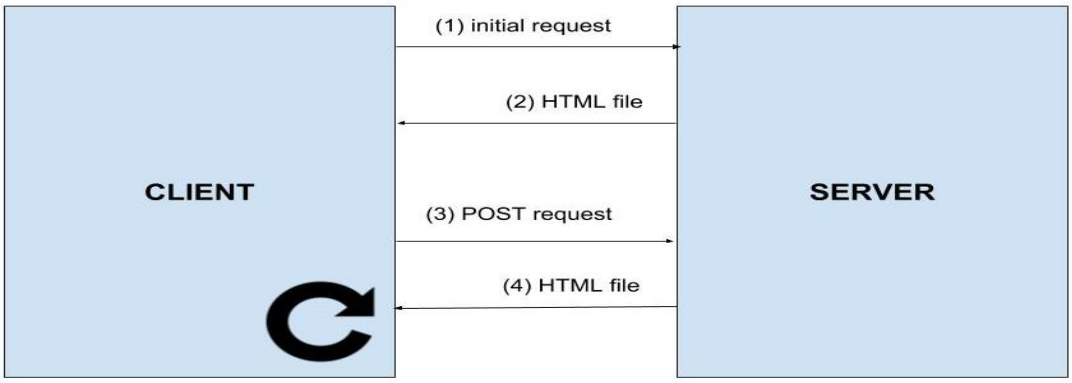

Fig. 2 - the multi page web-application lifecycle

The user navigating to the application's web address sends a request to receive the application home page (1), the server processes the request and, in accordance with the operating logic, sends an HTML file to the client part to display information to the user (2), then when the client sends a request for data addition or goes through internal links (3) the server processes this information and sends a new HTML file (4) to the client-side application which leads to to reload current page to another one in user browser **(Fig. 2)**.

With this approach, there is nothing wrong with simple applications. But if there is a need to create a rich user interface, then the page might become very complex and be loaded with a lot of data and this data is first generated on the server side of the application then sent to the client side via the Internet. Because of this, the speed of page loading increases and the user experience with the application deteriorates. [3] A completely different approach to web application development is the use of AJAX, which appeared in the 2000s. When using AJAX in the page code, it is possible to transfer and retrieve the necessary data without reloading the page itself and not spending any unnecessary traffic. This technology allows you to integrate into a multi-page web application function with the help of MPA, which improves the responsiveness of the application.

Pros of the Multiple-Page Application:

- MPAprovides users with better visualization of the application. multi-level navigation is the main part of a multipage application.
- Since MPA consist of many pages they are easier to promote with keywords they can be set for each page separately, which at the moment is the best solution for.SEO [4]
- Today it is the most popular solution for developing web applications. This means that you have a wide selection of literature and tutorials.
- Cons of the multiple-page application:
- The business logic of multi-page applications is closely related to the client side and the server side, which does not allow changing the server part without affecting the client.
- The development becomes quite complex. The developer needs to use frameworks for either client and server side. This results in the longer time of application development. [4]
- If you do not use Ajax, the page will be rebooted each time when user clicked on the link on web site
- It takes a lot longer to develop mobile applications. In most cases, you will need to coding back-end from scratch.

Traditional web applications use cases

- Traditional web applications can be easily launched in browsers that do not support js..
- The application is very simple and basically designed to display read-only information
- Resources (Developers) in our team aren't familiar with JavaScript or TypeScript or Front-end development. [5]

3. Single page application

A SPA is a web application that literally has one page - a single HTML page is loaded in the browser and is not reloaded during use. In other words, SPA is a web application hosted on a single web page that downloads all the necessary code for the job, along with the page itself. An application of this type appeared relatively recently, with the beginning of the era of HTML5 and SPA is a typical representative of applications on HTML5. The content of such a site is loaded from the server using AJAX - asynchronous JavaScript and XML. To implement the work through AJAX, you also need to implement part of the application on the server side. Typically, scripting languages are used. Since HTML is loaded on the client side, the size of the payload is reduced, because the server returns only JSON, not HTML [6].

Two possible implementations of the spa technology are available: all necessary code (html, css, js) is downloaded immediately to the client part of the application or dynamic loading of the required code in response to user operations.

SPAs request the markup (HTML) and data (JSON) independently and renders pages directly in the browser. The spa dynamically sends only data to the server, which takes less time and helps reduce the load on the network. This is done using asynchronous requests to the server using AJAX or using web sockets. **(Fig. 3)**.

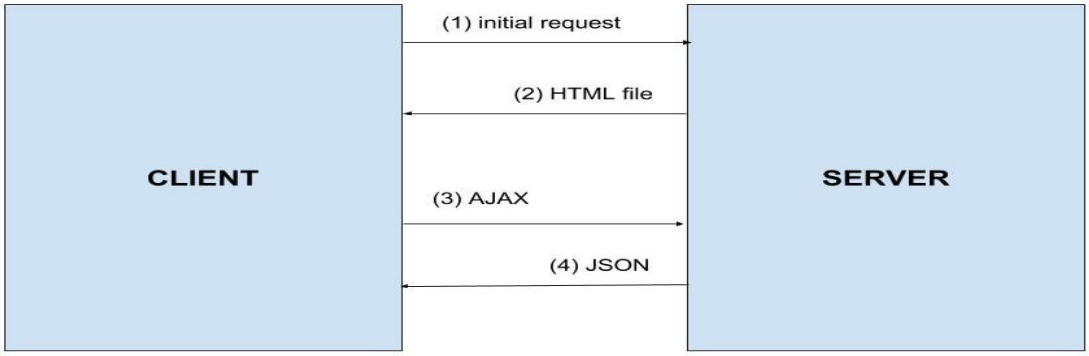

Fig. 3 - the single page application lifecycle

A single-page web application always works on the client side, so there is no constant reload of the page and the user can use the applications without any problems Every day millions of people use SPA, for example gmail, gitlab, facebook or google maps

Today, there are many frameworks and languages for building single-page web applications such as ReactJS, Angular3, VueJS, ASP.NET, etc..

SPA supports client navigation. All the "walking" of the user on the page-modules is unambiguously fixed in the navigation history, and the navigation is "deep", that is, if the user copies and opens a link to the internal module-page in another browser or window, he will go to the corresponding page.

SPA is hosted on one web page, so all the scripts and styles necessary for the site's work should be defined in one place of the project - on a single web page. Single-page applications loads all the scripts required to start the application when the web page is initialized.

In SPA, the number of files with scripts can reach several hundred, or even thousands. And "downloading all scripts" does not mean that all hundreds and thousands of files with scripts will be loaded immediately when the site is loaded. To solve the problem of loading a large number of scripts in the SPA, an API called AMD.

AMD implements the ability to download scripts on demand. That is, if you need 3 scripts for the "main page" of a single-page application, they will be downloaded before the program starts. And if the user clicks on another page of a single-page

application, for example, "About the program", the AMD principle will load the module (script + markup) only before moving to this page.

Pros of the Single-Page Application:

- SPA work on a large number of devices, and therefore, by creating one application, you get a much larger audience of users than using a standard approach.
- Application development is greatly simplified, you do not need to write code to render on the server.
- Single-page applications can easily be debugged in a web browser without using additional extensions. You can look at network operations and track code changes on the fly.
- SPA can cache any local storage effectively. An application sends only one request, store all data, then it can use this data and works even offline [4].
- Since the web page is one, it's much easier to build a rich, rich user interface.
- With the help of the SPA it is possible to develop huge sites for solving non-trivial tasksBetter user experience
- Low cost of support in a long term.

Cons of the Single-page Application:

- Heavy client frameworks which are required to be loaded to the client
- So far, not all search engines support page rendering on the client side, which does not allow implementing effective management SEO [7].
- Duplication of routers (in comparison with the classical approach).
- Since the application has one entry point, there is a risk that one error can lead to an inoperative state of the entire application.

The application of the laboratory of mechatronics and robotics of ONAPT is rather complicated, it contains rich functionality, has many pages for rendering, a lot of data is used and application is developed exclusively for the needs of the laboratory, there is no need to use seo optimisation, so it was decided to use the SPA methodology.

In the Mechatronics Robot Laboratory ONAFT application, 7 pages with different content are used, each page also has subsections, where it is also displayed different from the main content.

The main idea of the SPA is that the user is loaded one page, basically this is index.html. In the process of working with the application, the user navigates through internal links, sends or receives data, and the operation of the spa is performed in such a way that the user does not notice how the content is downloaded, since not the entire page is rebooted, but only its main part (highlighted in red). This gives a big advantage over MPA, since it reduces the response time of the application to user actions. The use of spa in the development makes it possible to develop a mobile application in the future, because the server part of the application is built on the basis of the RESTful API architecture.

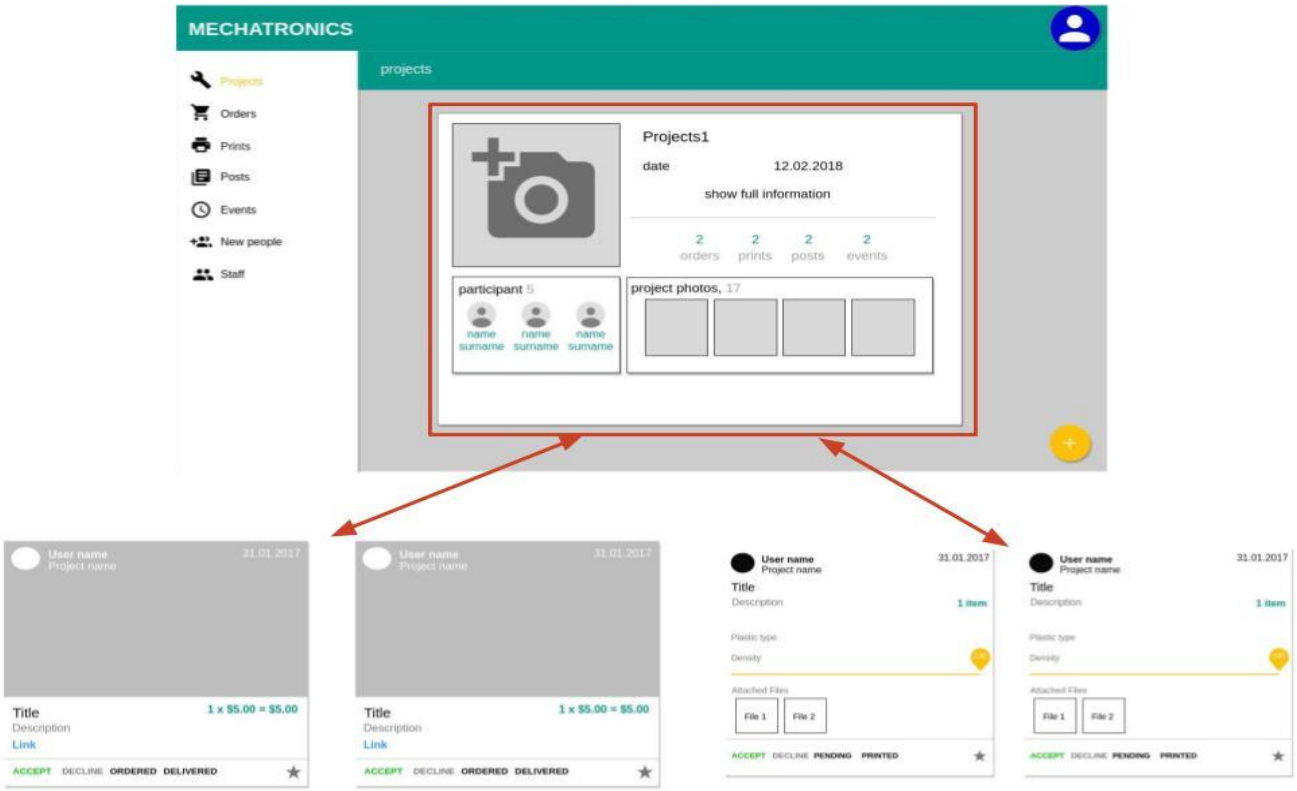

Fig. 4 - The Mechatronics Robot Laboratory ONAFT application

7. Conclusions

As more and more companies make their applications available on the Internet, some of them have huge functionality comparable to desktop applications. This provides a competitively comparable product on the market, relieves the user from the need to install applications on the desktop, the application will always be available on the Internet. Depending on the goals of the final product, you need to choose the development method, if the application is small, has several dynamic pages, and the main content is static, then it makes sense to select MPA. On the other hand, if the application has a complex structure, many volumes of data are wrapped up in it, a good UX is needed, then you should use the SPA. SPA more modern way for creating web applications, so this approach was chosen to create a Mechatronics Robot Laboratory ONAFT application application.

References

[1] Krunal (2008). Benefits of using the n-tiered approach for web applications. [online] Krunal-ajax-javascript.blogspot.bg. Available at: http://krunal-ajax-javascript.blogspot.bg/2008/09/benefits-of-using-n-tiered-approach-for.html [Accessed 5 Mar. 2018].

[2] Guryanov, A. (2018). WHAT'S THE DIFFERENCE BETWEEN SINGLE-PAGE APPLICATION AND MULTI-PAGE APPLICATION?. [online] www.adcisolutions.com. Available at: https://www.adcisolutions.com/knowledge/whats-difference-between-single-page-application-and-multi-page-application [Accessed 2 Mar. 2018].

[3] Shimanovsky, S. (2018). Multi page web applications vs. single page web applications. [online] Eikospartners.com. Available at: http://www.eikospartners.com/blog/multi-page-web-applications-vs.-single-page-web-applications [Accessed 27 Feb. 2018].

[4] Skólski, P. (2016). Single-Page Application vs. Multiple-Page Application - Neoteric. [online] Neoteric. Available at: https://neoteric.eu/single-page-application-vs-multiple-page-application [Accessed 1 Mar. 2018].

[5] Koukia, A. (2017). Going Single Page App or Traditional Web App – Aram Koukia. [online] koukia.ca. Available at: https://koukia.ca/going-single-page-app-or-traditional-web-app-cedb10041b50 [Accessed 1 Mar. 2018].

[6] Boyd, M. (2018). Single Page Applications: A Powerful Design Pattern for Modern Web Apps. [online] Eikospartners.com. Available at: http://www.eikospartners.com/blog/single-page-applications-a-powerful-design-pattern-for-modern-web-apps [Accessed 3 Mar. 2018].

[7] Lewis, J. (2017). I build advanced web applications for clients that appreciate design. [online] Josh Lewis. Available at: https://joshcarllewis.com/articles/single-page-applications-explained [Accessed 4 Mar. 2018].

Література

[1] Benefits of using the n-tiered approach for web applications. [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: http://krunal-ajax-javascript.blogspot.bg/2008/09/benefits-of-using-n-tiered-approach-for.html

[2] WHAT'S THE DIFFERENCE BETWEEN SINGLE-PAGE APPLICATION AND MULTI-PAGE APPLICATION? [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.adcisolutions.com/knowledge/whats-difference-between-single-page-application-and-multi-page-application.

[3] Shimanovsky, S. Multi page web applications vs. single page web applications. [Електронний ресурс] / Shimanovsky, S. – 2018. – Режим доступу до ресурсу: http://www.eikospartners.com/blog/multi-page-web-applications-vs.-single-page-web-applications.

[4] Skólski P. Single-Page Application vs. Multiple-Page Application [Електронний ресурс] / Skólski. – 2016. – Режим доступу до ресурсу: https://neoteric.eu/single-page-application-vs-multiple-page-application.

[5] Koukia A. Going Single Page App or Traditional Web App [Електронний ресурс] / Koukia. – 2017. – Режим доступу до ресурсу: https://koukia.ca/going-single-page-app-or-traditional-web-app-cedb10041b50.

[6] Boyd M. Single Page Applications: A Powerful Design Pattern for Modern Web Apps [Електронний ресурс] / M. Boyd. – 2018. – Режим доступу до ресурсу: http://www.eikospartners.com/blog/single-page-applications-a-powerful-design-pattern-for-modern-web-apps.

[7] Lewis J. I build advanced web applications for clients that appreciate design [Електронний ресурс] / Lewis. – 2017. – Режим доступу до ресурсу: https://joshcarllewis.com/articles/single-page-applications-explained.