UDC 004.051
DOI https://doi.org/10.32851/tnv-tech.2022.3.7

DESIGN OF WEB-APPLICATIONS IN THE CONTEXT OF OPTIMIZING THEIR PERFORMANCE

Slabinoha M. O. *– Ph.D. in Engineering,*
Associate Professor at the Department of Computer Systems and Networks
Ivano-Frankivsk National Technical University of Oil and Gas
ORCID ID: 0000-0002-7296-0356
Scopus Author ID: 57283728000

Chaban S. V. *– Student at the Department of Computer Systems and Networks*
Ivano-Frankivsk National Technical University of Oil and Gas
ORCID ID: 0000-0001-5830-7046

The purpose of this paper is to highlight recommendations for designing web-based software to maximize the speed of rendering the user interface and confirm these recommendations with the developed web-application.

The subject of this paper is very important, because web applications consume more and more resources both on the client side and server side. The Web became the large dumping ground for digital waste, and instead of saving the planet from increasing pollution through digitalization processes, it constantly demands production of more powerful digital devices for clients and highload server systems for data servers, which leads to more physical waste as a result. The aim to reduce the amount of computing where this is possible should be the primary goal for each software developer, including developers of web applications. That's why the concept of sustainable web development is so important right now to the whole of humanity.

This paper provides analysis of the subject area, comparative characteristics of the comparative characteristics of approaches to application design and formation of performance criteria, definition of recommended approaches, introduces the tools for the example web application implementation, and designing process for a web application based on previously defined approaches and testing its performance.

The methods of designing web-oriented software are improved with separation of logic and mapping components, taking into account the recommendations for improving the performance of the web application. This allows to achieve high performance of the web application.

The practical importance of the paper is the development of recommendations for approaches to the design of web-based software, which can then be used in the design of other web applications.

Key words: *sustainable web development, web applications, performance optimization, software development, client-server systems.*

Слабінога М. О., Чабан С. В. Розробка веб-додатків в контексті оптимізації їх швидкодії

Метою даної статті є виділення рекомендацій щодо проектування веб-додатків для максимальної швидкості відображення інтерфейсу користувача та розробка веб-додатку, що буде базуватися на основі даних рекомендацій з метою підтвердження їх ефективності.

Проблематика статті є актуальною, оскільки веб-додатки споживають все більше ресурсів як на стороні клієнта, так і на стороні сервера. Мережа стала великим звалищем цифрових відходів, і замість того, щоб врятувати планету від зростаючого забруднення через процеси цифровізації, вона постійно вимагає виробництва потужніших цифрових пристроїв для клієнтів і високонавантажених серверних систем для серверів даних, що в результаті призводить до збільшення забруднення через утилізацію застарілих пристроїв. Основною метою кожного розробника програмного забезпечення, включаючи розробників веб-додатків, має бути скорочення обсягу обчислень у додатках, де це можливо. Ось чому концепція веб-розробки в контексті сталого розвитку зараз настільки важлива для всього людства.

У цій роботі наведено аналіз предметної області, проведено порівняльний аналіз підходів до проектування веб-орієнтованих додатків та сформовано критерії їх ефективності. Також визначено рекомендовані підходи до розробки веб-додатків, вибрано інструменти для розробки тестового веб-додатку, а також висвітлено процес проектування веб-додатка на основі попередньо визначених підходів та тестування його ефективності.

Вдосконалені методи проектування веб-орієнтованого програмного забезпечення за рахунок розділення компонентів логіки та відображення, з урахуванням рекомендацій щодо підвищення продуктивності веб-додатку, що дозволяє досягти вищої швидкодії.

Практичне значення статті полягає у розробці рекомендацій щодо підходів до проектування веб-додатків, які потім можуть бути використані у веб-розробці.

Ключові слова: *веб-розробка в контексті сталого розвитку, веб-додаток, оптимізація продуктивності, розробка програмного забезпечення, клієнт-серверні системи.*

Introduction. Web development is a complex, long and multi-stage process. There are too many different approaches to web application development today. Many beginners, as well as experienced engineers, delve into modern and popular technologies, but forget about the fundamentals of how the web works. As a result, developers choose the wrong foundation for their web applications, which can greatly affect the quality of the final product.

One of the main decisions that web developers have to make is where to implement logic and mapping in their program. This can be difficult because there are several many ways to create a website. To better understand the architectures to choose from when making a decision, you need to have a clear understanding of each approach and the agreed terminology that can be used. The differences between these approaches help illustrate the trade-offs in rendering web applications through the prism of performance. That is why the task of creating methods and approaches to improve the performance of web applications at the design stage is relevant.

Therefore, the aim of the research was to develop recommendations for the design of web-based software in order to maximize the speed of rendering the user interface, which will be confirmed by the development of the application according to these recommendations.

Main part of the research. One of the most popular tools for estimating web page load speeds is Google PageSpeed Insights. PageSpeed Insights (PSI) reports on page performance on both mobile and desktop devices, as well as suggestions on how to improve the page. PSI provides both laboratory and field data about the page. Laboratory data is useful for troubleshooting performance because it is collected in a controlled environment. However, PSI does not take into account real critical places. Field data is useful for obtaining real-world user experience, but it has a more limited set of metrics. PSI also provides scores on other categories, such as SEO optimization, accessibility, and validation of PWA (Progressive Web Application) criteria.

It should be noted that PSI provides very general information about the speed of loading web pages and developers, in general, should look at these results as one of the metrics for building a website ranking in search engines, because this tool was designed for this purpose . When debugging web applications, you need to make more use of the developer's built-in browser tools (Performance and Network tabs) and other additional tools that will help you find the source of the problem.

However, it is not necessary to completely abandon PSI in terms of debugging a web application, as it provides useful information about key web metrics that can be used as a good place to start finding the problems themselves.

Key web metrics are a common set of performance signals that are important to any web experience. The main indicators of Web Vitals are FID, LCP and CLS, and they can be summarized at the page or source level.

Largest Contentful Paint (LCP). The download speed of the main content (titles, text, images, videos, etc.) measures how productively the download is performed, namely the time of the longest render of a text block or image visible in the preview area, counted from the start of page loading. To ensure user convenience, the LCP should be within 2.5 seconds of starting to load the page.

First Input Delay (FID). The time it takes to first interact with the content measures the interactivity of the webpage, namely the time from when the user first interacts with the page (ie, when he clicks a link, clicks a button, or uses a JavaScript-based control) until the browser actually will be able to begin to respond to signals from event handlers in response to this interaction. To ensure user convenience, the FID of pages should not exceed 100 milliseconds.

Cumulative Layout Shift (CLS). Aggregate web page layout offset measures visual stability. Unexpected movement of page content usually occurs due to asynchronous loading of resources or dynamic addition of DOM elements to the page on top of existing content. This may be due to an image or video of unknown size, a font that appears larger or smaller than its backup, third-party ads, or resizable widgets. To ensure the convenience of users, the CLS should not exceed 0.1.

Let's consider four main types of web-application user interface generation. Those are server-side rendering, static rendering, client-side rendering and hydration rendering.

Server-side rendering (SSR) generates a full-fledged server-side web page when the user navigates between web pages. This avoids additional requests for client-side data, as they are processed before the browser receives a response.

Server mapping typically creates a quick first Paint and a first Contentful Paint. Executing web page logic and server-side mapping avoids sending large amounts of JavaScript to the client, which helps to achieve a fast time to interactivity of the web page (Time to Interactive).

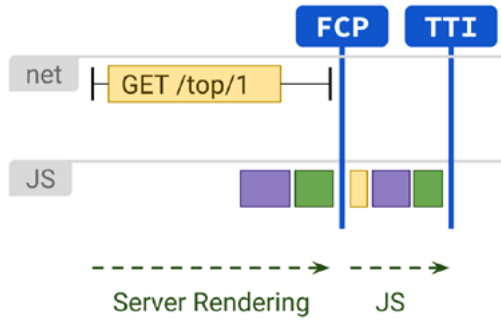

Figure 1. Server-side rendering scheme

Static rendering is the simplest of all web application render methods. A static web page is rendered during application build and provides fast first paint, fast first contentful paint, and time to interactivity, provided the number of JSs on the client side is limited. Unlike server-side visualization, static mapping can achieve consistently fast time to the first byte (TTFB) because HTML for a page does not need to be generated on the fly (runtime). Typically, a static rendering means creating a separate HTML file for each URL in advance. Because HTML responses are pre-generated, static visualizations can be deployed on multiple CDNs to take advantage of caching.

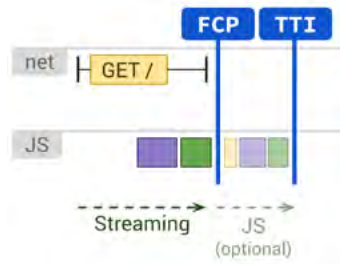

Figure 2. Static rendering scheme

Client-side rendering (CSR) means the reproduction of pages directly in the browser using JavaScript. All logic, data sampling, patterning, and routing are handled on the client, not the server.

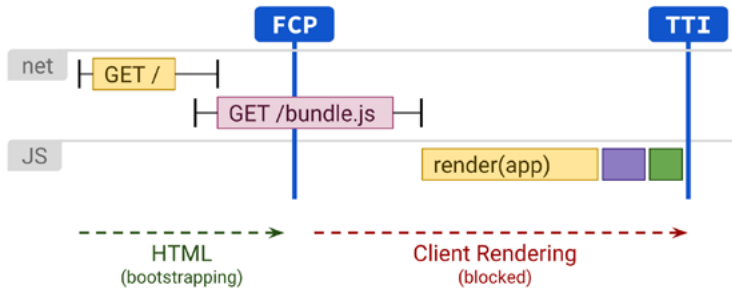

Figure 3. Client-side rendering scheme

Hydration, often referred to as universal or simply SSR, seeks to bridge the gap between client-side mapping and server-side mapping by doing both. Navigation requests, such as fully loading or reloading a page, are processed by a server that plays the web application in HTML, then JavaScript and the data used for rendering are embedded in the resulting document.

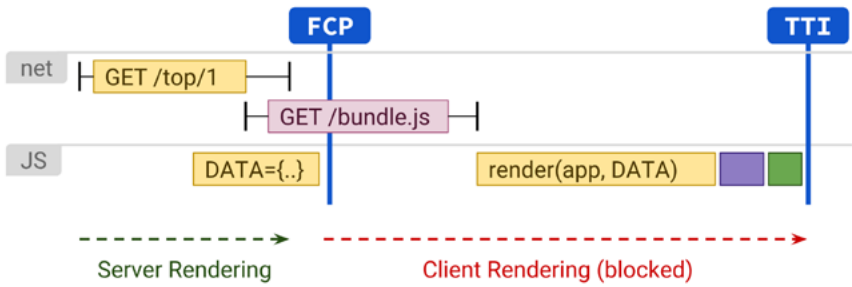

Figure 4. Hydration rendering scheme

Table 1 represents the advantages and disadvantages of each rendering approach. Each approach has its own pros and cons, as well as purpose of usage - from small HTML landing pages to heavy CRM applications and web-apps, so the question of choosing

the right rendering approach completely depends on web-development team decision and should be based on aim of the web-app, it's architecture, audience and other factors that affect the size, complexity and scale of the developed application.

Table 1

Advantages and disadvantages of different rendering approaches

Approach	Advantages	Disadvantages
Static rendering	Fast initial rendering and time to interaction Friendly to search engine optimization Incremental static regeneration	Poor scalability Lack of dynamic content Some users can see old version of pages due to cache issues
Client side rendering	Fast loading and navigation Single page app user experience	Slow initial rendering Bad for the search engine optimization Hard to optimize the performance, javascript bundle size is often too large
Server Side rendering	Fast initial rendering Friendly to search engine optimization Good browser support	Lack of dynamic content Most of the rendering work is done on server so you need powerful server systems when you have a lot of users
Hydration	Fast initial rendering Friendly to search engine optimization Dynamic content Streaming rendering	Practically, you need to build two applications - server side and client side You need to receive the data before you show the content

The main recommendations on building fast effective web-interface are:

1) Use a CDN. Content Delivery Network (CDN) is a geographically distributed network infrastructure that allows you to optimize the delivery and distribution of content to end users on the Internet. The use of CDN helps to increase the download speed of audio, video, software and other digital content by Internet users in the presence of CDN;

2) Caching static and media files. Today, there are many browser APIs that allow you to easily cache different types of client-side files, especially useful for web applications, which are mostly displayed on the client side using JavaScript. Service Workers allow you to cache static files with automatic revalidation. The Cache API provides a wider range of capabilities for caching and controlling different types of media files;

3) Do not display content that the user does not see. If you do not display images, videos and other heavy content that are not displayed in the user's display area, this will greatly increase the speed of the initial display of the web page;

4) Using Web Workers. If you have to perform some difficult calculations when displaying web pages, you should use Web Workers to take the load off the user interface;

5) Preload heavy pages. If you have JavaScript-heavy web pages, you can do additional optimization by pre-loading heavy scripts with simpler, static web pages;

6) Use Lazy Loading. Lazy loading allows you to break the code of a web application into JavaScript into pieces and download the necessary parts only when the user makes

a request. This greatly speeds up the first download of the web application, as the browser does not need to wait until the entire bundle of the web application is loaded;

7) Choose the correct location of servers. It often happens that developers have one server in one region and another server in a remote region from the first. This means that the user has to wait much longer for a response, so it is recommended that you place your servers and databases as close to each other as possible.

Taking into account these recommendations, the decision has been made to build the web application that will follow them. Projecthub is a web application that gives developers a centralized place to manage various aspects of their applications, such as logging changes, receiving customer feedback, a roadmap, and more.

This project includes a variety of web pages, both public and private, and requires a flexible approach to displaying different web pages, so this project is a good example of developing a web application with different approaches to web page architecture.

Project was built using the technologies Next.js, Node.js, PostgreSQL, Prisma, Railway and Supabase. Example of project page (search results page) and corresponding requests list is shown on figure 5.

The result of PSI metrics is shown on figure 6. This result demonstrates that the recommendations on building the web application are useful and help to improve web application performance and user experience. Worth mentioning that none of the pages has shown PSI score less than 92 points out of 100.

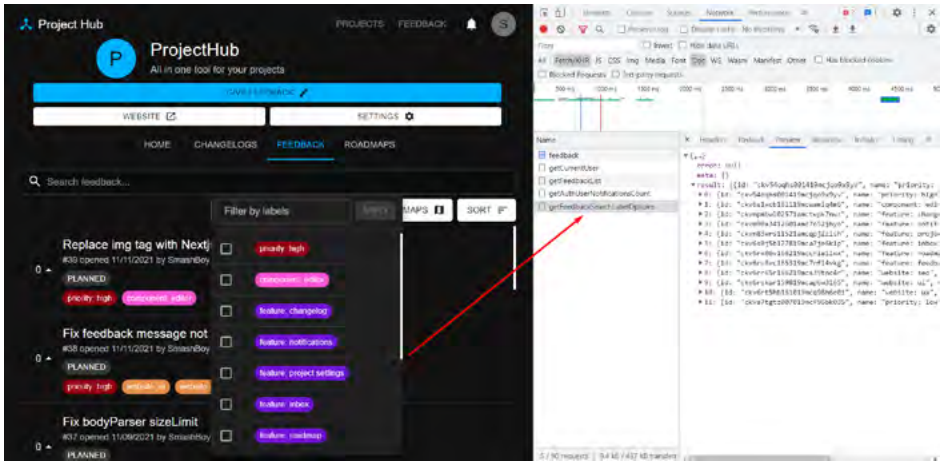

Figure 5. Example of user interface and requests list

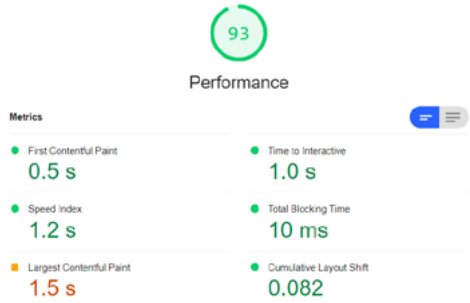

Figure 6. Result of PSI metrics for one of the Projecthub's pages

Conclusions. As it might be seen from the obtained metrics, the recommendations given during research allow to get good results when measuring the rendering performance. As a conclusion, we need to say that any of the architecture can reach high scores of rendering performance if web-developers will follow the mentioned recommendations while building the web-application from scratch. However, the "Software as a service" website builders like Wix or CMS's like Wordpress are not giving us as web-developers to change something in web application architecture, so the problem of their performance optimization is open and should be discussed separately.

Acknowledgements. *Authors want to thank the Armed Forces of Ukraine and all the defenders of Ukraine that give us the possibility to proceed scientific and engineering work in time of war.*

BIBLIOGRAPHY:

1. Gerry McGovern. World Wide Waste: How Digital Is Killing Our Plane and What We Can Do About It. Silver Beach, 2020. 171 p.
2. Онлайн ресурс Web Dev: веб-сайт. URL: https://web.dev (дата звернення: 25.05.2022).
3. Iskandar, Taufan Fadhilah, et al. Comparison between client-side and server-side rendering in the web development. In: IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2020. p. 21-36.
4. NAKANO, Yuusuke, et al. Web performance acceleration by caching rendering results. In: 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2015. p. 244-249.
5. BEKE, Mathias. On the Comparison of Software Quality Attributes for Client-side and Server-side Rendering. 2018. PhD Thesis. Department of Mathematics and Computer Science of the Faculty of Sciences, University of Antwerp.

REFERENCES:

1. McGovern, G. (2020). World Wide Waste: How Digital Is Killing Our Planet and What We Can Do About It. Silver Beach.
2. Let's build the future of the web, together. (2022). *Web.dev*. Retrieved May 25, 2022, from https://web.dev/.
3. Iskandar, T. F., Lubis, M., Kusumasari, T. F., & Lubis, A. R. (2020, May). Comparison between client-side and server-side rendering in the web development. In IOP Conference Series: Materials Science and Engineering (Vol. 801, No. 1, p. 012136). IOP Publishing.
4. Nakano, Y., Kamiyama, N., Shiomoto, K., Hasegawa, G., Murata, M., & Miyahara, H. (2015, August). Web performance acceleration by caching rendering results. In 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS) (pp. 244-249). IEEE.
5. Beke, M. (2018). On the Comparison of Software Quality Attributes for Client-side and Server-side Rendering (Doctoral dissertation, Department of Mathematics and Computer Science of the Faculty of Sciences, University of Antwerp).