UDC 004.773:004.738.52
DOI https://doi.org/10.32782/tnv-tech.2025.2.5

THE IMPACT OF SERVER-SIDE RENDERING ON SEO, USER EXPERIENCE AND PERFORMANCE IN WEB APPLICATIONS BUILT WITH ANGULAR

Borovskova Ye. A. *– Master's Degree, Software Engineer*
AppsFlyer Ltd
ORCID ID: 0009-0008-5481-8090

Server-side rendering (SSR) and Lazy Loading are key technologies for improving web application performance, user experience, and search engine optimisation. The study's relevance is driven by the need to create fast, scalable, SEO-optimised web applications that meet modern performance requirements. As one of the leading frameworks, Angular provides the technical basis for implementing these technologies. However, their integration is accompanied by numerous technical challenges that require analysis and development of appropriate solutions. The paper aims to analyse the impact of server-side rendering and Lazy Loading on the performance and usability of Angular web applications and develop practical recommendations for their combination. To achieve this goal, the article uses experimental modelling methods, including comparing performance indicators in real-world application conditions, an analytical method for assessing the technical and architectural aspects of integration, and methods for comparative analysis of scenarios with and without Lazy Loading. It has been proved that server-side rendering allows the generation of static HTML on the server, which ensures fast page loading and improves search engines' content indexing. It has been found that the use of Lazy Loading allows the reduction of the initial bundle size to 700 KB, which significantly reduces the first-page load time to 1 second in a 4G network and 1.5 seconds in a 3G network. It has been found that combining these technologies helps improve performance even on devices with low technical characteristics while reducing the load on server resources. The study identifies that the main challenges for implementation are the complexity of setting up server rendering to work in the client environment and the need to integrate caching mechanisms and optimise dynamic content. Recommendations for implementing hybrid approaches have been developed, including using server-side rendering for critical page elements and Lazy Loading for loading secondary components. It is recommended that caching mechanisms, performance monitoring tools, and priority rendering strategies be used to optimise system performance. Prospects for further research include the development of new approaches to integrating server-side rendering and Lazy Loading, particularly in the field of dynamic content, studying their impact on large distributed systems, and using artificial intelligence to automate rendering processes. This opens up opportunities for creating more adaptive and efficient web applications that meet the requirements of the modern digital environment.

Key words: *server-side rendering, Lazy Loading, performance, search engine optimisation, Angular, web applications, user experience.*

Боровскова Є. А. Вплив серверного рендерингу на SEO, користувацький досвід та продуктивність у веб-застосунках, розроблених за допомогою Angular

Серверний рендеринг (SSR) та Lazy Loading є ключовими технологіями для покращення продуктивності, користувацького досвіду та пошукової оптимізації веб-застосунків. Актуальність дослідження зумовлена потребою створення швидких, масштабованих та SEO-оптимізованих веб-застосунків, що відповідають сучасним вимогам до якості роботи. Angular, як один із провідних фреймворків, забезпечує технічну базу для впровадження цих технологій, однак їх інтеграція супроводжується численними технічними викликами, які потребують аналізу та розробки відповідних рішень.

Метою роботи є аналіз впливу серверного рендерингу та Lazy Loading на продуктивність і зручність використання веб-застосунків Angular, а також розробка практичних рекомендацій для їх поєднання. Для досягнення мети використано методи експериментального моделювання, що включають порівняння показників продуктивності у реальних умовах роботи застосунків, аналітичний метод для оцінки технічних і архітектурних аспектів інтеграції, а також методи порівняльного аналізу сценаріїв із використанням та без використання Lazy Loading.

Доведено, що серверний рендеринг дозволяє генерувати статичний HTML на сервері, що забезпечує швидке завантаження сторінок та покращує індексацію контенту пошуковими системами. Виявлено, що застосування Lazy Loading дає змогу зменшити розмір початкового бандлу до 700 КБ, що суттєво скорочує час першого завантаження сторінок до 1 секунди в 4G-мережі та до 1,5 секунди в 3G-мережі. З'ясовано, що поєднання цих технологій сприяє підвищенню продуктивності навіть на пристроях із низькими технічними характеристиками, одночасно знижуючи навантаження на серверні ресурси.

У ході дослідження визначено, що основними викликами для впровадження є складність налаштування серверного рендерингу для роботи в клієнтському середовищі, необхідність інтеграції механізмів кешування та оптимізації динамічного контенту. Розроблено рекомендації щодо впровадження гібридних підходів, які включають використання серверного рендерингу для критичних елементів сторінки та Lazy Loading для завантаження другорядних компонентів. Рекомендовано застосовувати механізми кешування, інструменти моніторингу продуктивності та стратегії пріоритетного рендерингу для оптимізації роботи системи.

Перспективи подальших досліджень передбачають розробку нових підходів до інтеграції серверного рендерингу та Lazy Loading, зокрема у сфері динамічного контенту, дослідження їх впливу на великі розподілені системи та застосування штучного інтелекту для автоматизації процесів рендерингу. Це відкриває можливості для створення більш адаптивних і ефективних веб-застосунків, які відповідають вимогам сучасного цифрового середовища.

Ключові слова: *серверний рендеринг, Lazy Loading, продуктивність, пошукова оптимізація, Angular, веб-застосунки, користувацький досвід.*

Problem statement. Server-side rendering, as one of the key technologies in web development, has become widespread due to its advantages in ensuring fast page loading, improving search engine optimisation, and increasing the usability of web applications. However, using server-side rendering in combination with Angular raises several issues related to the complexity of setting up the architecture, the need to optimise server resources, and ensuring uninterrupted operation under high loads. This problem is of great practical importance, as the efficiency of web applications used in various fields, including e-commerce, educational platforms, and media resources, depends on its solution.

From a scientific point of view, the issue of optimising server-side rendering in Angular projects requires a detailed study of its impact on performance and scalability. An important task is to study the mechanisms that allow for achieving a balance between system performance, end-user convenience, and compliance with modern SEO standards. These aspects determine the competitiveness of web applications and, therefore, are relevant for the further improvement of web development technologies.

Analysis of the latest research and publications. The impact of server-side rendering on SEO, user experience, and performance in web applications developed with Angular is a hot research topic covering indexing, performance, and resource optimisation.

The study by K. Kowalczyk and T. Szandala compares single-page and multi-page web applications regarding SEO. It pays attention to the key role of server-side rendering in improving search engine indexing. The authors note that search algorithms may partially or completely ignore dynamic pages without server-side rendering [1]. Angular Universal, as analysed by B. Borggreve, stands out as an effective tool for server-side rendering. The study underscores its capacity to generate HTML on the server prior to delivery, leading to faster rendering speeds, improved SEO performance, and reduced loading times, which collectively enhance user engagement [2].

In the field of progressive web applications, M. Hajian's research highlights the synergy between these applications and server-side rendering. Angular's architecture is identified as a key factor in delivering not only exceptional performance but also crucial SEO advantages, vital for optimising modern web solutions [3].

H. A. Jartarghar and his team turned their focus to the Next.js framework, presenting an in-depth examination of its server-side rendering capabilities in conjunction with React. Their research highlights its role in minimising page load times and improving the user experience while discussing the challenges of integrating server-side rendering into scalable projects [4].

Rendering speed and resource optimisation for large-scale web projects were central themes in C. L. Phang's comprehensive review. The study evaluates Angular's effectiveness in achieving these goals, making it a practical choice for developers handling extensive web applications [5].

Optimisation techniques for page loading, such as lazy loading, were explored by R.-M. Bâra, C. A. Boiangiu, and C. Tudose. Their work demonstrates the advantages of these methods for Angular applications, particularly in scenarios involving substantial data volumes [6].

D. C. Sathyakumar explored scalable applications, with a specific emphasis on resource optimisation strategies. The study introduces methods for leveraging server-side rendering to ensure rapid and seamless content delivery while conserving server resources [7].

Addressing enterprise-level requirements, V. De Sanctis examined the adaptation of Angular Universal for reducing computing costs and enhancing user satisfaction. The research underscores the value of server-side rendering in streamlining resource allocation and boosting application performance [8].

A detailed exploration of modern rendering approaches was conducted by A. Bampakos and M. Thompson, who provided practical insights into the implementation of server-side rendering in real-world projects. Their findings emphasise the flexibility of Angular Universal in addressing SEO and user experience challenges [9].

Dynamic web application development within the ASP.NET Core ecosystem was the focus of O. Gumus and M. T. S. Ragupathi. Their work explores the integration of server-side rendering with Angular, showcasing its ability to ensure cross-platform compatibility and reduced loading times [10].

J. Wilken and S. Seshadri provided foundational insights into Angular's capabilities for server-side rendering. Their studies highlight the necessity of optimising web applications to meet modern performance standards, focusing on speed and efficiency [11, 12].

The comparative effectiveness of rendering methods within Next.js is analysed by R. Hanafi, A. Haq, and N. Agustin. Their research reveals the substantial reduction in content display delays achieved through server-side rendering, which significantly enhances the user experience [13].

Practical applications of Angular Universal were detailed by A. Aristeidis and M. Thompson. They showcase how this tool contributes to the development of high-performance web applications capable of handling heavy traffic loads [14].

The intersection of server-side rendering and information security was explored by K. Chyzhmar, O. Dniprov, O. Korotiuk, R. Shapoval, and O. Sydorenko. Their study underlines the role of server rendering in mitigating risks and fortifying data protection in web applications [15].

Legal implications of web technologies, particularly in the context of electronic jurisdiction, were examined by O. Kostenko and V. Their analysis highlights the performance and interaction benefits of server rendering, which holds significance for legal and technical frameworks [16]. Thus, the reviewed studies confirm that server-side rendering in Angular web applications helps to improve SEO, increase performance,

and provide an effective user experience. Each work makes a unique contribution to understanding these aspects, offering different approaches to solving the problems of modern web applications.

Highlighting previously unsolved parts of the problem. Despite the achievements in implementing server-side rendering in Angular applications, important aspects of optimising its technical and architectural solutions to improve performance and ensure SEO efficiency remain unresolved. In particular, the impact of server-side rendering on the indexing of content by search engines and the specific benefits of this technology for improving the ranking of web applications in search queries have not been sufficiently studied.

Additional attention should be paid to the problems of combining server-side rendering with Lazy Loading. This technology provides optimised resource loading, but its integration into server-rendered Angular applications causes difficulties due to the complexity of settings and the impact on the initial project structure. These challenges are especially acute on devices with limited technical characteristics, where it is critical to minimise page load times.

The technical challenges of integrating server-side rendering into Angular projects remain relevant. These include compatibility issues with popular libraries, the lack of unified approaches to settings, and limited tools to solve these problems.

The proposed research aims to analyse modern approaches to server-side rendering, study its interaction with Lazy Loading, and develop practical recommendations for their effective combination. This will reduce technical limitations, improve the performance of web applications, and ensure their compliance with modern SEO and usability standards.

The purpose of the article is to study the impact of server-side rendering on search engine optimisation, user experience, and performance of web applications developed using Angular and to develop recommendations for its effective use in the context of modern web technology requirements.

Research objectives:

1. To analyse modern approaches to implementing server-side rendering in Angular web applications, focusing on technical and architectural features and assessing its impact on content indexing by search engines and SEO benefits.

2. Investigate the impact of Lazy Loading on the performance of Angular web applications, in particular on reducing the initial bandwidth and page load time and optimising resources for devices with low technical characteristics. Also, investigate the challenges of integrating this technology with server-side rendering.

3. To develop practical recommendations for combining server-side rendering and Lazy Loading in Angular web applications to improve performance, usability and compliance with modern SEO standards, considering technical challenges.

Summary of the main material. Server-side rendering (SSR) plays a critical role in enhancing the performance and scalability of Angular web applications. Various modern approaches have been developed to address the challenges of implementing SSR, including the integration of Angular Universal, hybrid rendering, and static site generation (SSG). These methods not only optimise the rendering process but also cater to specific use cases, such as dynamic content handling or improved initial page load performance. Each approach requires distinct architectural and technical considerations, such as infrastructure support or application structure analysis. Table 1 presents a detailed comparison of these approaches, focusing on their implementation features, benefits, and challenges, providing insights into their applicability in different scenarios.

Table 1

Comparison of modern approaches to implementing server-side rendering in Angular web applications

Approach	Implementation features	Benefits and challenges
Angular Universal	Node.js rendering, integration with existing Angular applications	Improved SEO and first load speed, but requires additional server infrastructure
Hybrid-rendering	Combining SSR and CSR to optimise performance	Fast rendering and load adaptability, but complex configuration
Priority rendering	Identify key elements for rendering and downloading priority data	Reduces server load, but requires careful analysis of the web application structure
Static site generation (SSG)	Pre-render pages during project build	Maximum download speed for static content, but limited for dynamic content

Source: compiled by the author based on [3, 9, 14].

In practice, server-side rendering is used to solve specific tasks in various scenarios. In modern projects, Angular Universal allows you to quickly integrate SSR into existing web applications by automating the creation of HTML pages. For example, large online stores are actively using SSR to improve the indexing of products by search engines and reduce page load times. Hybrid rendering has found applications in platforms where fast interface response is critical, such as online learning platforms. However, implementing server-side rendering requires significant resources to set up and maintain the infrastructure, which is challenging for small and medium-sized companies.

In search engine optimization (SEO), server-side rendering (SSR) offers unique advantages by providing static HTML to search engines and reducing page load times. Unlike traditional client-side rendering, SSR enables faster indexing of dynamic content and enhances the relevance of web pages in search engine results through optimized meta descriptions, titles, and structured data. These features improve visibility, higher search rankings, and better user engagement. Table 2 highlights the specific factors impacted by SSR about SEO, showcasing its ability to optimize content indexing, reduce bounce rates, and ensure an overall seamless user experience.

Table 2

The impact of server-side rendering on content indexing and the main benefits for SEO

Factor	The impact of server rendering	Main benefits for SEO
Indexing of dynamic content	Providing static HTML pages to search engines	Increase the volume of indexed content
Page load time	Generate pages on the server before transferring them to the client	Reduce the time of first display and improve search engine rankings
Using meta tags	Generate optimised meta descriptions, titles, and structured data on the server	Increase relevance and visibility in search engines
User experience	Provide quick access to the main content of the page	Reduced bounce rate due to better user experience

Source: compiled by the author based on [2, 3, 5, 11].

Server-side rendering is widely used today to ensure efficient content indexing in a competitive search engine environment. Large platforms, such as e-commerce or media resources, use server-side rendering to optimise their content for the requirements of search algorithms. For example, by generating HTML on the server, companies can include the necessary meta tags for each page, which makes it easier for search engines to identify them [5]. In addition, the fast display of the pages' main content helps increase the time users spend on the site, which also positively impacts search engine rankings. At the same time, the implementation of server-side rendering requires significant resources to set up the infrastructure. However, a significant increase in SEO efficiency and website visibility in search engines justifies these costs.

Lazy Loading is an important technology affecting modern web application performance. Its principle is to load only those modules or components that the user needs when performing a certain action, while delaying the Loading of the rest [6]. This approach helps reduce the initial bundle's size and significantly reduce page loading time, which is especially important for devices with low technical characteristics or a slow Internet connection.

To evaluate the effectiveness of Lazy Loading, an experiment was conducted on a 10-page web application created using Angular. The study analysed two scenarios: loading a web application without lazy loading and loading it with it. In each case, the following indicators were evaluated: the size of the initial bundle, page loading time in networks with different connection speeds, and device resource usage (Table 3).

Table 3

The impact of Lazy Loading on the performance of Angular web applications

Parameter	No Lazy Loading	With Lazy Loading
Initial size of the bundle	~2.5 MB for a website with 10 pages	~700 KB due to downloading only the required pages
Download time (4G network)	~3 seconds	~1 second
Download time (3G network)	~5 seconds with high memory usage	~1.5 seconds with minimal resource requirements
RAM usage	High, especially on devices with low specifications	Low, reducing the load on devices

Source: author's own development

Experimental results show that Lazy Loading significantly improves the performance of Angular web applications. When using Lazy Loading, the size of the initial bundle is reduced by almost four times, significantly reducing page load times. Using the example of a 4G connection, the loading time was reduced from 3 seconds to 1 second, and for a 3G connection, it was reduced from 5 seconds to 1.5 seconds. This is especially important for users with low-end devices, as Lazy Loading significantly reduces the use of RAM and CPU time.

Today, Lazy Loading is widely used in multi-page web applications to improve performance. For example, in large online stores or information portals, this technology allows you to load only the functionality that the user needs, avoiding wasted resources. The Angular implementation, based on the dynamic Loading of modules through the loadChildren directive, is an effective solution for optimising application performance. This approach provides a better user experience and allows developers to create scalable, productive systems that meet the requirements of the modern web environment.

Server-side rendering, implemented in Angular projects using Angular Universal, is an effective approach to ensure fast page loading and improve content indexing by search engines. However, this process is accompanied by several technical challenges that affect web applications' performance, stability, and flexibility. One of the key challenges is the complexity of setting up server-side rendering, which requires additional efforts to synchronise server and client code. Angular Universal requires developers to adapt the application to a server-side environment, sometimes modifying existing components, eliminating dependencies on browser APIs, and using specific libraries.

Another problem is processing dynamic content that depends on server requests. To ensure correct operation, you need to set up caching or pre-rendering mechanisms in advance, which increases the complexity of the project architecture. Another challenge is ensuring rendering stability under high loads, as each request requires significant computing resources on the server. This can be a critical issue for large, high-traffic projects, as server-side rendering significantly increases the load on server resources compared to client-side rendering.

Combining server-side rendering and Lazy Loading in Angular web applications is a powerful approach to ensure high performance, usability, and compliance with modern SEO standards [11]. Server-side rendering with Angular Universal allows you to generate static HTML on the server, which speeds up the first Loading of pages and ensures their availability for indexing by search engines. Lazy Loading, in turn, helps to optimise the Loading of resources, allowing you to distribute the load during the application's operation and avoid loading unnecessary content at the initial stage.

The practical combination of these two approaches requires careful application architecture planning. Initially, server-side rendering should be used to generate the basic HTML code that includes key page elements such as titles, meta descriptions, and visible content needed for quick display. This will ensure optimal conditions for SEO and improve the first user experience. At the same time, Lazy Loading can be integrated to dynamically load modules that are not critical for the first rendering, such as UI elements that will only be needed after user interaction.

An important element is to configure Angular routing to support Lazy Loading. This involves using dynamic Loading of modules through the `loadChildren` directive, which avoids overloading the server and the client device. In addition, integrating caching mechanisms for server-side rendering can significantly reduce the time it takes to re-generate HTML, providing faster request processing. Using priority rendering strategies also allows you to concentrate server resources on loading the page's most important elements.

Implementing this approach requires considering the specifics of client and server environments. For example, you should ensure that server rendering is compatible with dynamic elements that are loaded via Lazy Loading. This may involve using performance monitoring tools to track load times and page correctness and conducting testing to assess the impact on user experience. The combination of server-side rendering and Lazy Loading allows you to create high-performance web applications that meet modern SEO standards while providing fast access to content and optimal use of client device resources.

Conclusions and prospects for further research. A study of the impact of server-side rendering and Lazy Loading on the performance of Angular web applications has shown that combining these technologies significantly improves page loading speed, reduces the amount of resources used, and improves the quality of user experience. Server-side rendering ensures the generation of HTML code on the server, which

positively impacts SEO performance, particularly due to better indexing of content by search engines. Lazy Loading, in turn, minimises the size of the initial bundle and optimises the performance of applications on devices with low technical characteristics.

The study identified the main problems as the high complexity of setting up server rendering for integration with client code, the need to optimise dynamic content, and the limitation of server resources during high loads. Although Lazy Loading is effective for performance, it requires careful routing configuration and analysis of module dependencies to ensure stable operation.

Based on the results obtained, hybrid approaches that combine server-side rendering and Lazy Loading are recommended to optimise the performance of Angular web applications. Server content caching mechanisms and strategies for priority rendering of key elements should be implemented to reduce loading time and resource costs. Integrating performance monitoring tools to detect and resolve technical issues promptly is also important.

Prospects for further research include the development of innovative architectural approaches to improve the interaction of server rendering and Lazy Loading, particularly in the field of dynamic content. A separate area of research is the analysis of the impact of these technologies on the operation of large distributed systems with high loads, as well as the study of the possibilities of using artificial intelligence to automate rendering optimisation processes in web applications.

BIBLIOGRAPHY:

1. Kowalczyk K., Szandala T. Enhancing SEO in Single-Page Web Applications in Contrast With Multi-Page Applications. *IEEE Access*. 2024. Vol. 12. P. 11597–11614. DOI: https://doi.org/10.1109/ACCESS.2024.3355740 (date of access: 11.12.2024).

2. Borggreve B. Server-Side Enterprise Development with Angular: Use Angular Universal to pre-render your web pages, improving SEO and application UX. Packt Publishing Ltd, 2018. URL: https://books.google.com.ua/books?id=Wt18DwAAQBAJ (date of access: 11.12.2024).

3. Hajian M. Progressive Web Apps with Angular. Apress Berkeley, CA. 2018. URL: https://link.springer.com/book/10.1007/978-1-4842-4448-7 (date of access: 11.12.2024).

4. Jartarghar H.A., Salanke G.R., A.R A.K., G.S S., Dalali S. React Apps with Server-Side Rendering: Next.js. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*. 2022. Vol. 14(4). P. 25–29. DOI: https://doi.org/10.54554/jtec.2022.14.04.005 (date of access: 11.12.2024).

5. Phang C.L. Mastering Front-End Web Development (HTML, Bootstrap, CSS, SEO, Cordova, SVG, ECMAScript, JavaScript, WebGL, Web Design and many more). Chong Lip Phang, 2020. URL: https://books.google.com.ua/books?id=Y-UJEAAAQBAJ (date of access: 11.12.2024).

6. Bâra R.-M., Boiangiu C.A., Tudose C. Analysing the performance impacts of lazy loading in web applications. *Journal of Information Systems & Operations Management*. 2024. Vol. 18(1). P. 1–15. URL: https://web.rau.ro/websites/jisom/Vol.18%20No.1%20-%202024/JISOM%2018.1.pdf#page=9 (date of access: 11.12.2024).

7. Sathyakumar D.C. Techniques and Practices for Optimizing Resources in Large Scale Horizontal Web Applications That Deliver Cross Functional UX Components. *2024 IEEE International Conference on Electro Information Technology (eIT)*. Eau Claire, WI, USA, 2024. P. 468–479. DOI: https://doi.org/10.1109/eIT60633.2024.10609896 (date of access: 11.12.2024).

8. De Sanctis V. ASP.NET Core 5 and Angular: Full-stack Web Development with. NET 5 and Angular 11. Packt Publishing Ltd, 2021. URL: https://books.google.com.ua/books?id=Dr0YEAAAQBAJ (date of access: 11.12.2024).

9. Bampakos A., Thompson M. Angular Projects: Build modern web apps by exploring Angular 12 with 10 different projects and cutting-edge technologies. Packt Publishing, 2021. URL: https://ieeexplore.ieee.org/document/10163614 (date of access: 11.12.2024).

10. Gumus O., Ragupathi M. T. S. ASP.NET Core 2 Fundamentals: Build cross-platform apps and dynamic web services with this server-side web application framework. Packt Publishing Ltd, 2018. URL: https://books.google.com.ua/books?id=9kZsDwAAQBAJ (date of access: 11.12.2024).

11. Wilken J. Angular in Action. Simon and Schuster, 2018. URL: https://books.google.com.ua/books?id=XTgzEAAAQBAJ (date of access: 11.12.2024).

12. Seshadri S. Angular: Up and Running: Learning Angular, Step by Step. O'Reilly Media, Inc., 2018. URL: https://books.google.com.ua/books?id=CAFeDwAAQBAJ (date of access: 11.12.2024).

13. Hanafi R., Haq A., Agustin N. Comparison of Web Page Rendering Methods Based on Next.js Framework Using Page Loading Time Test. *Teknika*. 2024. Vol. 13(1). P. 102–108. URL: https://ejournal.ikado.ac.id/index.php/teknika/article/view/769 (date of access: 11.12.2024).

14. Aristeidis B., Thompson M. Angular Projects: Build Modern Web Apps by Exploring Angular 12 with 10 Different Projects and Cutting-Edge Technologies. Packt Publishing Ltd, 2021. URL: https://books.google.com.ua/books?id=Dr0YEAAAQBAJ (date of access: 11.12.2024).

15. Chyzhmar K., Dniprov O., Korotiuk O., Shapoval R., Sydorenko O. State Information Security as a Challenge of Information and Computer Technology Development. Journal of Security and Sustainability Issues. 2020. Vol. 9(3). P. 819–828 URL: https://www.researchgate.net/publication/340545387_STATE_INFORMATION_SECURITY_AS_A_CHALLENGE_OF_INFORMATION_AND_COMPUTER_TECHNOLOGY_DEVELOPMENT (date of access: 11.12.2024).

16. Kostenko O., Furashev V. Genesis of Legal Regulation Web and the Model of the Electronic Jurisdiction of the Metaverse. Bratislava Law Review. 2022. Vol. 6(2). P. 21–36. DOI: https://doi.org/10.46282/blr.2022.6.2.316 (date of access: 11.12.2024).

REFERENCES:

1. Kowalczyk, K., & Szandala, T. (2024). Enhancing SEO in single-page web applications in contrast with multi-page applications. *IEEE Access, 12*, 11597–11614. https://doi.org/10.1109/ACCESS.2024.3355740

2. Borggreve, B. (2018). *Server-side enterprise development with Angular: Use Angular Universal to pre-render your web pages, improving SEO and application UX*. Packt Publishing Ltd. Retrieved from https://books.google.com.ua/books?id=Wt18DwAAQBAJ

3. Hajian, M. (2018). *Progressive web apps with Angular*. Apress. https://doi.org/10.1007/978-1-4842-4448-7

4. Jartarghar, H. A., Salanke, G. R., A. R., A. K., G. S., S., & Dalali, S. (2022). React apps with server-side rendering: Next.js. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 14*(4), 25–29. https://doi.org/10.54554/jtec.2022.14.04.005

5. Phang, C. L. (2020). *Mastering front-end web development (HTML, Bootstrap, CSS, SEO, Cordova, SVG, ECMAScript, JavaScript, WebGL, Web Design and many more)*. Chong Lip Phang. Retrieved from https://books.google.com.ua/books?id=Y-UJEAAAQBAJ

6. Bâra, R.-M., Boiangiu, C. A., & Tudose, C. (2024). Analysing the performance impacts of lazy loading in web applications. *Journal of Information Systems & Operations Management, 18*(1), 1–15. Retrieved from https://web.rau.ro/websites/jisom/Vol.18%20No.1%20-%202024/JISOM%2018.1.pdf#page=9

7. Sathyakumar, D. C. (2024). Techniques and practices for optimizing resources in large scale horizontal web applications that deliver cross functional UX components. *2024 IEEE International Conference on Electro Information Technology (eIT)*, 468–479. https://doi.org/10.1109/eIT60633.2024.10609896

8. De Sanctis, V. (2021). *ASP.NET Core 5 and Angular: Full-stack web development with. NET 5 and Angular 11*. Packt Publishing Ltd. Retrieved from https://books.google.com.ua/books?id=Dr0YEAAAQBAJ

9. Bampakos, A., & Thompson, M. (2021). *Angular projects: Build modern web apps by exploring Angular 12 with 10 different projects and cutting-edge technologies*. Packt Publishing Ltd. Retrieved from https://ieeexplore.ieee.org/document/10163614

10. Gumus, O., & Ragupathi, M. T. S. (2018). *ASP.NET Core 2 Fundamentals: Build cross-platform apps and dynamic web services with this server-side web application framework*. Packt Publishing Ltd. Retrieved from https://books.google.com.ua/books?id=9kZsDwAAQBAJ

11. Wilken, J. (2018). *Angular in action*. Simon and Schuster. Retrieved from https://books.google.com.ua/books?id=XTgzEAAAQBAJ

12. Seshadri, S. (2018). *Angular: Up and running: Learning Angular, step by step*. O'Reilly Media, Inc. Retrieved from https://books.google.com.ua/books?id=CAFeDwAAQBAJ

13. Hanafi, R., Haq, A., & Agustin, N. (2024). Comparison of web page rendering methods based on Next.js framework using page loading time test. *Teknika*, *13*(1), 102–108. Retrieved from https://ejournal.ikado.ac.id/index.php/teknika/article/view/769

14. Aristeidis, B., & Thompson, M. (2021). *Angular projects: Build modern web apps by exploring Angular 12 with 10 different projects and cutting-edge technologies*. Packt Publishing Ltd. Retrieved from https://books.google.com.ua/books?id=Dr0YEAAAQBAJ

15. Chyzhmar, K., Dniprov, O., Korotiuk, O., Shapoval, R., & Sydorenko, O. (2020). State information security as a challenge of information and computer technology development. *Journal of Security and Sustainability Issues, 9*(3), 819–828. https://www.researchgate.net/publication/340545387_STATE_INFORMATION_SECURITY_AS_A_CHALLENGE_OF_INFORMATION_AND_COMPUTER_TECHNOLOGY_DEVELOPMENT

16. Kostenko, O., & Furashev, V. (2022). Genesis of legal regulation web and the model of the electronic jurisdiction of the metaverse. Bratislava Law Review, 6(2), 21–36. https://doi.org/10.46282/blr.2022.6.2.316