UDC 004.41
DOI DOI DOI DOI DOI https://doi.org/10.32782/2663-5941/2026.2.1/42

Ushakova I.O.
https://orcid.org/0000-0001-8315-0917
Simon Kuznets Kharkiv National University of Economics

MODERNIZATION OF THE BACKEND OF THE WEB APPLICATION BASED ON MICROSERVICE ARCHITECTURE

The article considers the problem of inefficiency of web applications built on the basis of monolithic architecture, in particular in terms of speed, scalability and support. The relevance of the study is due to the increasing load on information systems and the need to ensure high performance when working with a large volume of user requests. The purpose of the work is to justify the feasibility of transitioning from a monolithic architecture of the backend of the web application to a microservice architecture and increasing the efficiency of the system through its structural decomposition. In the process of the study, an analysis of the existing architecture of the application was carried out, its logical and functional components were determined, which allowed for the decomposition of modules and the transition to separate independent services. The main functional microservices were identified , which are responsible for working with users, authorization, processing text, graphic and video data, working with vacancies and resumes, search engines, social networks, payment systems, mail and SMS services, as well as a security and administration module. The proposed architectural solution allowed to ensure independent scaling of individual services, simplify the process of their modernization and optimize the distribution of computing resources. The study contains a comparative analysis of the performance indicators of the web application before modernization and after the implementation of the microservice architecture. The results of testing the functions of administration, search and integration with social networks are presented, which indicate a significant improvement in the system response time. It is shown that the average increase in speed is from several times to several dozen times depending on the type of request. Special attention is paid to the optimization of the cluster infrastructure by redistributing resources and integrating specialized data storage and processing services. The results obtained confirm the effectiveness of using the microservice architecture to improve the server part of web applications and can be used in the development and modernization of complex information systems focused on high load, flexibility and stability of operation.

Keywords: *web application, backend, monolithic architecture, microservice architecture, structural decomposition, cluster infrastructure, performance testing.*

Formulation of the problem. Software architecture plays a key role in the software development process. Software architecture approaches are principles for organizing the structure of a system to ensure its reliability, scalability, and ease of maintenance. The choice of architecture primarily depends on the scale of the project: monolithic architecture is better for small applications and is suitable for small businesses, while microservices are more suitable for large, complex systems. Also, when choosing an architectural approach, the requirements for system flexibility, constraints on the duration of development, the qualifications of the development team, and the scale of the project are taken into account.

When choosing the right approach, software developers often do not take into account future risks or problems that they may encounter in the future. It is the problem of underestimating architectural solutions that can play a key role in the quality of a software product. Therefore, the problem of choosing the right approach to defining software architecture is relevant and requires a comprehensive analysis and justification of approaches to its solution. In addition, there is the problem of technology obsolescence, when software was developed using outdated technologies and therefore does not meet the requirements of modern technologies. In general, owners of such software applications are most often dissatisfied with the speed of the software, its scalability and support.

Microservice architecture allows you not to be limited to one technology, but to develop software applications using different technologies. This architecture allows you to divide the software product into modules that can be written in different program-

ming languages, which also allows you to minimize future problems, dynamically change development approaches and methods to improve the quality of the software product.

Therefore, development using microservice architecture is very promising for implementation, this architecture has been most actively developing recently in the field of software development. The main advantages of such a solution, compared to monolithic architecture and serverless, are the savings of resources, both human and material – during the development process, and computational – during the scaling process. However, it is also worth considering the difficulties that arise when using microservices, mainly related to the complication of transaction mechanisms, as well as the need to implement systems for automated monitoring and logging of program operation. Therefore, the microservice architecture of software development itself was chosen for further development.

The purpose of the work is to justify the feasibility of transitioning from a monolithic architecture of the backend of a web application to a microservice architecture and to increase the efficiency of the system's functioning through its structural decomposition. This will contribute to increasing the speed of the application, reducing the cost of technical resources and expanding their functional capabilities. To achieve the goal, the following tasks must be solved in the work: to analyze the issues of modern server-side software architectures; to review and analyze existing methods for building software architecture; to analyze the functionality and perform structural decomposition of the web application; to improve the web application based on microservice architecture and to analyze the effectiveness of the new architectural solution.

Analysis the last research and publications. The problem of choosing the right approach to building a web application architecture A significant number of scientific works are devoted to it. It is relevant and requires a comprehensive analysis and development of approaches to its solution. It is noted that microservices architecture is fundamentally different from traditional monolithic architecture, where a software system essentially consists of a single deployable system that encompasses all the functions of the software system [3, 5–7, 11]. This important distinction makes microservices modular, independently scalable, and technologically diverse compared to monoliths.

In the work of Järvinen K. emphasizes that the main principles of microservices include decomposition into bounded contexts, independent deployment, and decentralized management. Explicitly limiting responsibilities within clear boundaries makes microservices flexible, easy to maintain, and allows teams to work independently of other teams without excessive communication. The author explores how the advantage in flexibility of microservices depends on their scalability. The paper uses empirical and quantitative methodology to analyze different levels of scalability of microservices and their impact on performance and costs of cloud technologies [2].

Paper «From Monolithic Systems to Microservices» notes that microservices implementation still faces challenges in terms of time and computational resources, scalability, orchestration, organization, etc. [11]. There are procedures for migrating from monolithic architectures to microservices, but none of them accurately quantify the performance difference. The paper proposes an approach to evaluate performance and the relationship between different application variables to compare monolithic and microservice architectures.

The problem of shortcomings of existing approaches to microservice architecture, especially for systems operating in dynamic real-time conditions, is raised in the article «Microservices Architecture for Real-Time» [5]. In this study, the authors propose a new scalable microservices architecture optimized for real-time data processing using a modular, event-driven design. The paper describes a system that can consume real-time weather information based on data available through the OpenWeatherMap application interface. The system is built using complementary technologies used in modern data processing and microservices architectures, such as Apache Kafka, Apache Flink, Redis, Kubernetes, and based on adaptive autoscaling through KEDA and HPA in the architecture.

Recently, the number of studies on cloud microservices has been increasing. In the pape of Sigala N. S. examines the architecture of microservices from the perspective of practical issues related to the processes of its design, deployment, and management in cloud computing environments [9]. The results presented in this study provide a deep understanding of how to effectively implement and manage microservices architectures to make software solutions more efficient, reliable, and scalable in cloud computing environments.

In the article by Mendonça N. C., Box C., Manolache C., Ryan L. notes that despite the advantages of microservices, their implementation poses some technical and organizational challenges related to opera-

tional complexity, increased technological diversity, and the need for better coordination between teams [4]. The authors describe design decisions, tradeoffs, and lessons learned from the open source service network project Istio, which had a microservices architecture from the beginning. However, less than three years after the first release of Istio, problems arose and the control plane microservices were merged into a monolith.

Different approaches to security can significantly influence architectural decisions. A monolithic system interacts with the external environment as a single unit and can provide simpler access control. The security of a monolith is provided centrally, simplifying protection but creating the risk of a single point of failure. On the other hand, in microservices, each element can have its own security system, which provides more flexible but also more complex control. In the book Siriwardena P., Dias N. addresses the security issues of microservices [10]. The book covers the basics of securing both the application boundary and the communication between services. The authors explore how to deploy and secure microservices, access microservices from a single page, and web application.

Microservices require more investment in development, but this approach is justified in the following cases: high load on the application, which must stably serve thousands of users simultaneously; a long-term and large-scale project that will constantly develop and grow; complex business processes that are more convenient to divide into services and scale independently; large teams that can work in parallel on different services without conflicts in the code. That is, the problem of choosing the right approach to defining the software architecture requires a comprehensive analysis and development of approaches to its solution. To compare the characteristics of a web application built using different architectural approaches, performance testing is performed using various tools [12, 13]. In the work Selivorstova T., Klishch S., Kyrychenko S., Guda A., Ostrovskaya K. Analysis of monolithic and microservice architectures: a statistical analysis of the main quantitative characteristics of monolithic and microservice architectures was performed using the Kubectl tool for managing Kubernetes clusters. [6]. The authors use the Kubernetes cluster deployment methodology for microservice architecture using Minikube, Kubectl, and Docker. The results confirmed that the microservice architecture has much greater fault tolerance and flexibility compared to the monolithic architecture.

Task statement. The main tasks of the work are to improve the architecture of the backend of the software application and analyze the results of its performance. The improved system should be divided into modules that will work independently of each other and interact using the HTTP protocol. The application built on the basis of monolithic architecture will be improved. The basic system is a web application and has two interfaces: for candidates looking for work; for employers looking for candidates for vacant positions. Employers and candidates can post articles and advertisements. The application also has integration with third-party services, namely social networks and payment systems. To monetize the web application, a decision was made to develop special functionality that should interact with many parts of the system. Before the application was modernized, an analysis was conducted of the number of visits and the directions of its use by users. It turned out that the main problem was the speed of the application. This was a consequence of the fact that the web application was developed quite a long time ago using outdated and no longer supported technologies. To solve this problem, the task of changing the monolithic architecture to a microservice one was set.

Outline of the main material of the study

Decisions on improving the software application. Before improving the software application, a complete analysis of the existing system was carried out, which made it possible to divide it into logical and functional modules. The old system included the following services: user service; authorization service; social network service; article service; marketing service ; resume and vacancy service; search service; SMS and mail service; payment system service .

After analyzing the existing structure, work was carried out to create new and decompose old modules. The following main functional modules were identified: user management module; authorization module; image management module; video file management module ; text document management module; blog article management module; article comment management module; blog post moderation module; resume management module; job search module; resume search module; social media management module; payment system management module; mail service management module; SMS service management module ; security module; advertising management module.

The existing web application, in addition to the main system, also had an additional administration panel. This panel also required architectural changes to ensure correct interaction with the main system. From the very beginning, the web application had one

large monolithic service that managed all actions on the administrator panel. This architectural approach worsened the speed of the application; its structural diagram is presented in Fig. 1.

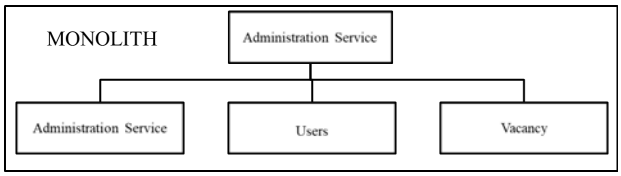

Fig. 1. Structural diagram of the monolithic architecture of the administration panel

As a result of refactoring, the following modules were identified in the system: user administration; vacancy administration; resume administration; search tag administration; statistics viewing. The structural diagram of the changed architecture of the administrator panel is presented in Fig. 2. Decomposition into modules allowed us to encapsulate the logic of the software product, which will speed up the speed of work in the future. Decomposition of the web application into modules allowed us to identify problematic modules, namely the resume and vacancy search module, CRUD queries for working with the administration panel, as well as modules for working with social networks and users. These modules had to be redesigned.

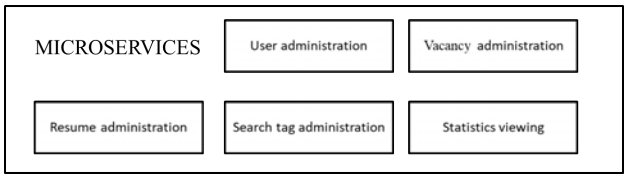

Fig. 2. Structural diagram of the microservice architecture of the administration panel

Thanks to the transition to microservice architecture, it became possible to write modules in different languages, so these modules were rewritten from Java to PHP, which has long proven itself both in terms of greater simplicity and speed of development, and in terms of increased speed of operation of the application itself, written in this language. In addition, the development team had the necessary skills. A module for monetizing the software application was also to be developed. This module should be independent and reusable in another software application. In addition to the software part, the cluster on which the web application was located also needed changes. The cluster consists of two database replications and eight computing servers. The structural diagram of the existing cluster is presented in Fig. 3.

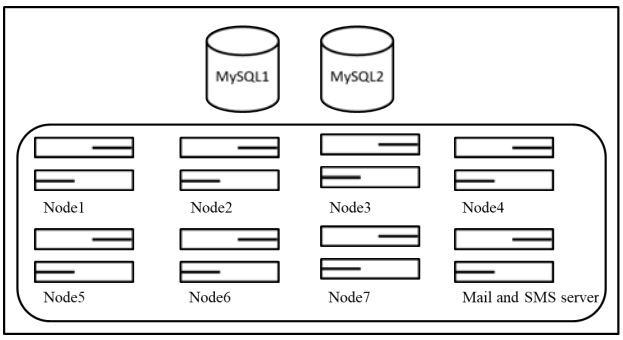

Fig. 3. Structural diagram of the old cluster

The main problem of the software product was the speed of work. Therefore, an analysis of the work of the created modules was carried out. As a result of the analysis, it was found that some modules require a larger amount of resources, and some, on the contrary, scarce resources. Therefore, a decision was made to optimize the cluster structure. Thanks to the redistribution of computing resources between the modules, it was possible to reduce the number of servers to three. Most of the functionality for searching in the web application was transferred to the ElasticSearch search server database, which is often used as a NoSQL database. Elasticsearch is an open-source distributed search and analytical engine built on the basis of the Apache Lucene library. It provides fast full-text search, storage and analysis of large volumes of data (Big Data) in real time using JSON documents. The search server has the following advantages: designed for instant search, scales from a single server to large clusters; has full-text search, which provides efficient search of text information, supports relevance, synonyms and analytics; thanks to the REST architectural style, the API interacts with data using standard HTTP requests, which makes it convenient for developers; it is often used for log analysis and monitoring together with Logstash for data collection and Kibana for visualization. This technology made it possible to store a large amount of data with quick access to it. Services for working with mail and SMS were also allocated. They were transferred to AWS services. The optimized cluster structure is presented in Fig. 4.

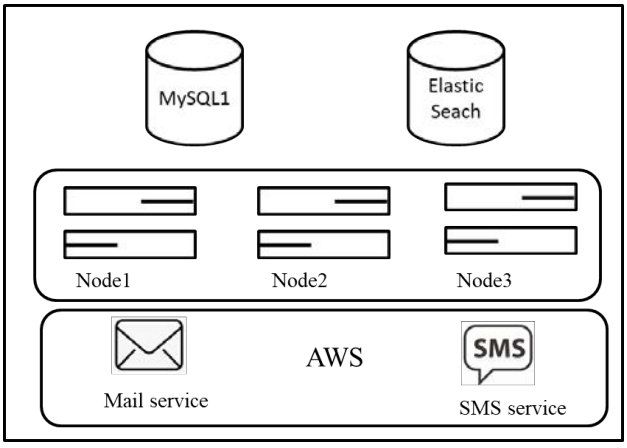

Fig. 4. Optimized cluster structure diagram

Comparative analysis of the effectiveness of a software application for monolithic and microservice architecture. To analyze the effective the microservice approach to web application architecture was tested with server response speed measurements. Fig. 4 shows the response speed measurements of the "Administration" service request in a web application on a monolithic architecture, and Fig. 5 shows measurements of the response speed of the "Administration" microservice request in a web application based on a microservice architecture.

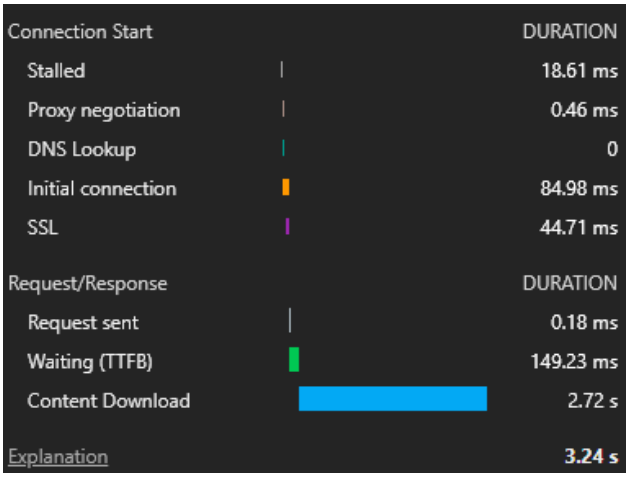

Fig. 5. Measurements of the response speed of the "Administration" service request in a web application on a monolithic architecture

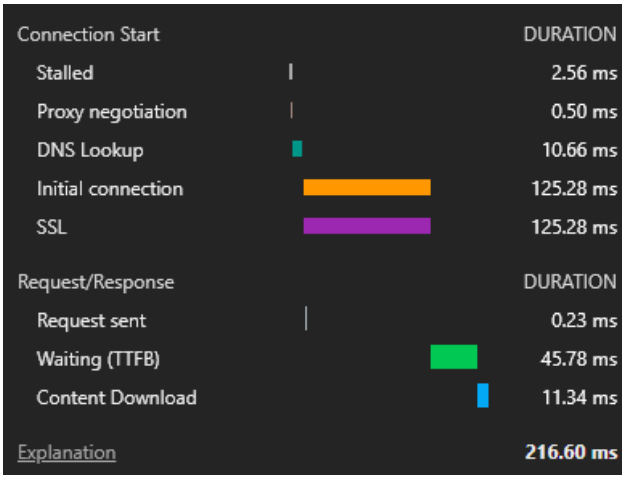

Fig. 6. Measurements of the response speed of the Administration microservice request in a web application on a microservice architecture

A comparative analysis of the results of testing the "Administration" function for monolithic and microservice architectures is presented in Table 1.

Fig. 6 shows measurements of the response speed of the "Vacancy Search" service request in a web application on a monolithic architecture, and Fig. 7 shows measurements of the response speed of the microservice request "Search for vacancies" in a web application based on a microservice architecture.

Table 1

Results of testing the "Administration" function using different server-side architectures

Indicator	Time, ms		Relative growth
	Monolithic architecture	Microservice architecture	
Resume list page	3410	216	15.79
User list page	2989	197	15.17
Job listing page	3018	203	14.87
Article list page	3704	300	12.34
Statistics page	2050	100	20.5

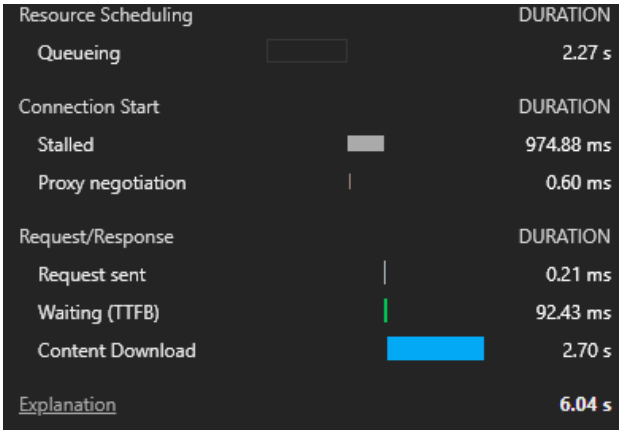

Fig. 7. Measurements of the response speed of the Job Search service request in a web application on a monolithic architecture

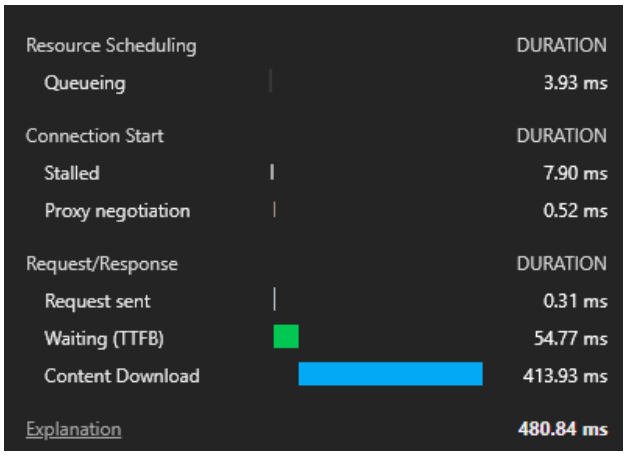

Fig. 8. Measurements of the response speed of the microservice request "Search for vacancies" in a web application on a microservice architecture

A comparative analysis of the results of testing the "Job Search" function for monolithic and microservice architectures is presented in Table 2.

Figure 8 shows measurements of the response speed of a service request for working with social networks in a web application on a monolithic architecture, and Fig. 9 shows measurements of the

response speed of a microservice request for working with social networks in a web application based on a microservice architecture.

Table 2

Results of testing the "Job Search" function using different server-side architectures

Indicator	Time, ms		Relative growth
	Monolithic architecture	Microservice architecture	
Search for vacancies without filters	2179	423	5.15
Search for vacancies with 1 filter	4902	458	10.7
Search for vacancies with 2 filters	6040	480	12.58
Search for vacancies with 4 filters	8298	503	16.09
Search for vacancies with 16 filters	17921	489	36.65

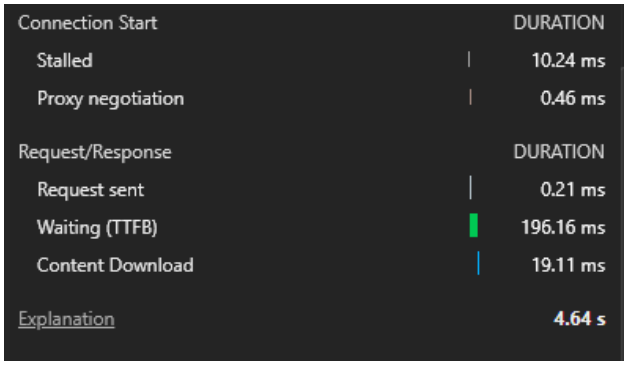

Fig. 9. Measurements of the response speed of a service request for working with social networks in a web application on a monolithic architecture

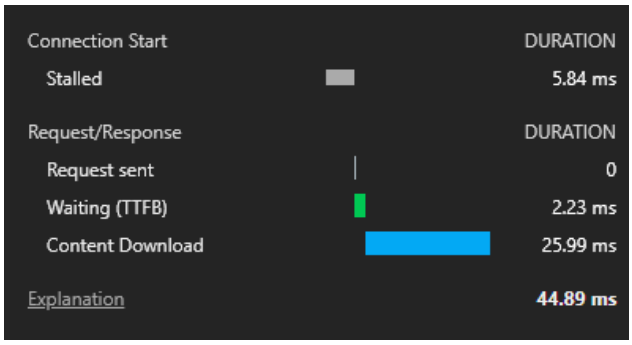

Fig. 10. Measurements of the response speed of a microservice request for working with social networks in a web application based on a microservice architecture

The results of testing the "Working with social networks" function for monolithic and microservice architecture are presented in Table 3. Analysis of the test results showed that the use of microservice architecture is more appropriate, because the service request response speed indicators are better than monolithic architecture. The average indicators of the tests performed showed a speed increase in the range from 3 to 36 times. These indicators may indicate that the system has been successfully improved and the decision to switch to a new type of architecture was correct.

Table 3

Results of testing the "Working with social networks" function using different server-side architectures

Indicator	Time, ms		Relative growth
	Monolithic architecture	Microservice architecture	
Authorization when working with Facebook	3580	1203	2.98
Data request when working with Facebook	2790	980	2.85
Data processing when working with Facebook	1246	44	28.31
Authorization when working with Google	2987	891	3.35
Data query when working with Google	1790	594	3.01
Data processing when working with Google	988	89	11.1
Authorization when working with Linkedin	2199	918	2.4
Data request when working with Linkedin	2123	702	3.02
Data processing when working with Linkedin	1008	76	13.26

It should also be noted that due to changes in the cluster architecture, both current and future financial costs have been significantly reduced. The main problem with the old approach was the problem of scaling, which took up too many hardware resources. Now the load can be easily distributed between different services, thereby making it a less expensive procedure.

Comparison of the analyzed methods. Development using microservice architecture is very promising for implementation. This architecture has been most actively developing recently in the field of software development. The main advantages of such a solution compared to monolithic architecture are the savings of resources, both human and material – during the development process, and computational – during the scaling process. However, it is also worth considering the difficulties that arise when using

microservices, mainly related to the complication of transaction mechanisms, as well as the need to implement systems for automated monitoring and logging of program operation. Therefore, the microservice architecture of software development itself was chosen for further development.

Conclusions. The article considers the task of improving the server part of a web application by moving from a monolithic architecture to a microservice architecture. The analysis showed that the monolithic approach, despite its simplicity of implementation, has significant limitations in terms of scalability, performance, and support, especially in the context of an increasing number of users and the complexity of the system's business logic.

As part of the study, a structural analysis of the existing software application was carried out and its decomposition into separate logical and functional modules was performed. Based on the results of this analysis, a microservice architecture was formed, which includes services for working with users, authorization and authentication, content management, processing text, graphic and video data, job search and resumes, integration with social networks, interaction with payment systems, as well as security and administration modules. This approach allowed us to clearly delineate the functional responsibilities of the system components, reduce their interdependence and create conditions for the independent development and updating of each service. An important stage of the work was the improvement of the architecture of the cluster infrastructure that ensures the functioning of the web application. The redistribution of computing resources between individual services allowed us to optimize the load on servers, reduce the number of physical or virtual machines and at the same time increase the speed of processing user requests. The integration of specialized tools for searching and processing large data sets helped reduce the load on the main database and ensured more efficient use of hardware resources. Separate services for processing e-mail and SMS messages were outsourced to external cloud services, which further increased the stability and scalability of the system.

A comparative analysis of the performance of monolithic and microservice architectures showed a significant increase in performance for the main functions of the web application. In particular, a reduction in response time was recorded for the administration, job search, and integration with social networks modules. The average values of the productivity increase ranged from several times to dozens of times, which confirms the effectiveness of the chosen approach. In addition to improving technical characteristics, the proposed architectural solution has important organizational and economic advantages. The decomposition of the system into separate services creates conditions for the parallel work of several development teams, which reduces the time to implement new functions and simplifies the process of supporting the software product. The ability to use different programming languages and technology stacks to implement individual components increases the flexibility of the system and allows you to choose the most appropriate tools for solving specific tasks.

The results obtained indicate the feasibility of using microservice architecture to modernize outdated web applications. The proposed solution allows you to increase the flexibility of the system, simplify its maintenance, and create conditions for further scaling without significant changes in the structure of the program code. In addition, the microservice approach provides the ability to use different technologies to implement individual components, which increases the system's adaptability to changing requirements and the emergence of new development tools.

The scientific significance of the study lies in substantiating practical approaches to refactoring the architecture of the server part of web applications and in forming methodological recommendations for the transition from monolithic solutions to microservices. The practical significance of the work lies in the possibility of using the obtained results in the design and optimization of real information systems focused on working with a large number of users and high performance requirements.

Thus, the study confirmed that the implementation of microservice architecture is an effective means of improving the server part of web applications, allows achieving a significant increase in performance, optimizing resource use and providing conditions for further development of the software product.

Bibliography:

1. Aydemir F., Başçiftçi F. Building a performance efficient core banking system based on the microservices architecture. *Journal of Grid Computing*. 2022. Vol. 20. No. 4. 37 p. Doi: 10.1007/s10723- 022-09624-z.

2. Järvinen K. Analysing the Effects of Scalability in Microservices to UserPerceived Performance and Cloud Costs. Master of Science (Tech) Thesis. 2025. 102 p.

3. Kalske M. Transforming monolithic architecture towards microservice architecture: Master of Science (Tech) Thesis. 2018. 76 p.

4. Mendonça N. C., Box C., Manolache C., Ryan L. Why Istio Migrated From Microservices to a Monolithic Architecture *IEEE Software Magazine*. 2021. №5. P. 1 – 22.

5. Reddy D. N., Suryodai R., Kumar S. V., Ambika M., Muniyandy E., Krishna6 R. V., Abdurasul B A Scalable Microservices Architecture for Real-Time. *International Journal of Advanced Computer Science and Applications (IJACSA).* Vol. 16. No. 9. 2025. P 31 – 42. DOI: https://dx.doi.org/10.14569/ IJACSA.2025.0160905.

6. Selivorstova T., Klishch S., Kyrychenko S., Guda A., Ostrovskaya K. Analysis of monolithic and microservice architectures: Features and metrics. *International scientific journal «Computer systems and information technologies»*. 2021. № 3. P. 59 – 60. Doi: 10.31891/CSIT-2021-5-8.

7. Shaik Sh. Technical review: Server-side Composition vs. Client-side Composition. *World Journal of Advanced Research and Reviews*. 2025. 26(02). P. 3037-3046.

8. Shethiya A. S. Scalability and Performance Optimization in Web Application Development. *Integrated Journal of Science and Technology Computer Science & Information Technology.* 2025. Vol. 2. Iss. 1. 7 p.

9. Sigala N. S. Microservices Architecture in Cloud Computing: A Software Engineering Perspective on Design, Deployment, and Management. *International journal of research and innovation in social science (IJRISS).* 2025. Vol. IX. Iss. XV. P. 215 – 239. Doi: https://dx.doi.org/10.47772/IJRISS.2025.915EC0013.

10. Siriwardena P., Dias N. Microservices Security in Action: Design secure network and API endpoint security for Microservices. *Manning Publications Co.* 2020. 616 p.

11. Tapia F., Mora M. Á., Fuertes W., Aules H., Aules H. Flores E., Toulkeridis Th. From Monolithic Systems to Microservices: A Comparative Study of Performance. *Applied Sciences*. 2020. №10 (17). 30 p. Doi:10.3390/ app10175797.

12. Ushakova I., Plokha O., Skorin Yu. Approaches to Web Application Performance Testing And Real-Time Visualization of Results. *Bulletin of Kharkiv National Automobile and Highway University.* 2022. №96. P.71 – 80.

13. Ushakova, I., Skorin, Y., Shcherbakov, A. Methods of quality assurance of software development based on a systems approach. *CEUR Workshop Proceedings*. 2021. № 3200. P. 158 – 168.

Ушакова І.О. МОДЕРНІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ ВЕБЗАСТОСУНКУ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

У статті розглянуто проблему неефективності функціонування вебзастосунків, побудованих на основі монолітної архітектури, зокрема в аспектах швидкодії, масштабованості та підтримки. Актуальність дослідження обумовлена зростанням навантаження на інформаційні системи та потребою забезпечення високої продуктивності при роботі з великим обсягом користувацьких запитів. Метою роботи є обґрунтування доцільності переходу від монолітної архітектури серверної частини вебзастосунку до мікросервісної архітектури та підвищення ефективності функціонування системи шляхом її структурної декомпозиції. В процесі дослідження проведено аналіз існуючої архітектури застосунку, визначено його логічні та функціональні компоненти, що дозволило здійснити декомпозицію модулів та перехід до окремих незалежних сервісів. Виділено основні функціональні мікросервіси, що відповідають за роботу з користувачами, авторизацію, обробку текстових, графічних та відеоданих, роботу з вакансіями та резюме, пошукові механізми, соціальні мережі, платіжні системи, поштові та SMS сервіси, а також модуль безпеки й адміністрування. Запропоноване архітектурне рішення дозволило забезпечити незалежне масштабування окремих сервісів, спростити процес їх модернізації та оптимізувати розподіл обчислювальних ресурсів. Дослідження містить порівняльний аналіз показників продуктивності вебзастосунку до модернізації та після впровадження мікросервісної архітектури. Наведено результати тестування функцій адміністрування, пошуку та інтеграції із соціальними мережами, які свідчать про суттєве покращення часу відповіді системи. Показано, що середній приріст швидкодії становить від кількох разів до кількох десятків разів залежно від типу запиту. Окрему увагу приділено оптимізації кластерної інфраструктури шляхом перерозподілу ресурсів та інтеграції спеціалізованих сервісів зберігання та обробки даних. Отримані результати підтверджують ефективність застосування мікросервісної архітектури для вдосконалення серверної частини вебзастосунків і можуть бути використані при розробленні та модернізації складних інформаційних систем, орієнтованих на високе навантаження, гнучкість та стабільність функціонування.

Ключові слова: *вебзастосунок, монолітна архітектура, мікросервісна архітектура, структурна декомпозиція, кластерна інфраструктура, тестування продуктивності.*

Дата першого надходження статті до видання: 17.02.2026
Дата прийняття статті до друку після рецензування: 13.03.2026
Дата публікації (оприлюднення) статті 11.05.2026