Journal of Theoretical and Applied Information Technology
31st March 2025. Vol.103. No.6
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN AND DEVELOPMENT OF MICROSERVICES-BASED CRM SYSTEM

ANDRII PRYHODA[1], ROSTYSLAV SIKORA[2], VOLODYMYR MOSKALENKO[3], ANDRII ROSKLADKA[4]

[1]State University of Trade and Economics, Department of Software Engineering and Cybersecurity, Ukraine
[2]State University of Trade and Economics, Department of Digital Economy and System Analysis, Ukraine
[3]state University of Trade and Economics, Department of Software Engineering and Cybersecurity, Ukraine
[4] State University of Trade and Economics, Department of Software Engineering and Cybersecurity, Ukraine
E-mail: [1]alesamytnyk@gmail.com, [2]sikorarostyslav@knute.edu.ua, [3]moskalenko878@knute.edu.ua, [4]roskladka.andrii@gmail.com

ABSTRACT

The aim of the article is to analyse the technical and organizational aspects of the implementation of microservice architecture in CRM systems. The research employed analytical and logical methods, technical analysis of microservice architecture comparing it with monolithic systems, as well as Docker and Kubernetes applied technologies. The use of Docker and Kubernetes containerization tools is shown to be a key factor in successfully managing a microservices-based CRM infrastructure. The analysis of implementation results shows how the microservice architecture reduces operational costs and increase the efficiency of system support. The role of automation of information systems (IS) in business was analysed. It was established that the modern market creates a situation where it is necessary to constantly increase production efficiency. General requirements for the CRM system were formulated, including security and protection, reliability and availability, user-friendly interface, ease of implementation and maintenance, availability of basic functionality. Current web application development technologies are explored and the technologies used in the project, including Django, Django Rest Framework, React, MobX, and Ant Design are described.

Keywords: *CRM system, Docker, Kubernetes, Microservice architecture, Monolithic architecture.*

1. INTRODUCTION

The latest ITs provide the best methods of information processing and analysis, as they significantly expand the possibilities of data manipulation. Automated information systems designed for accounting expand the professional competencies of specialists. Current market conditions require continuous improvement of productivity, prompt response to changes, improvement of customer service and reduction of losses. According to [1], the development of digital technologies resulted in the emergence of new approaches to the design of Customer Relationship Management (CRM) systems, which have led to changes in the planning and technical implementation stages. The CRM system is planned taking into account the constant saturation of data and large volumes, which is provided due to the microservice architecture. This requires a thorough analysis of the requirements for each module of the system, determination of their independence and opportunities for further expansion of functions. The technical implementation introduces the latest technologies of containerization, automation of CI/CD (Continuous Integration/Continuous Deployment), infrastructure management through orchestration systems. This approach enables developers to quickly make changes to individual microservices without the need to stop the entire system.

Creating a web application, especially one as complex as a CRM system, requires the use of certain technologies. Wang et al. [2] notes that a web application consists of a client part (which the user sees and interacts with in a browser window), and a server part (which is responsible for data processing). The Django framework implemented in Python was chosen as the basis of the server part of the programme. Django was created in 2003 as a result of the developers' activities engaged in the creation of websites, when they decided to combine the functionality for the rapid creation of websites in a particular framework. According to StackOverflow, almost 42% of professional developers use Python in their work, and 13% use Django as a platform for building backend applications as of 2020.

The physical part of the microservice architecture involves the use of independent modules that work as separate services, each having its own database, network interface, and computing resources. They are typically hosted in Docker containers and managed through orchestration systems such as Kubernetes. According to [3], the appropriate architecture enables individual components of the CRM system to function autonomously, which reduces dependence on the central core and minimizes the risks of a general system failure. Containerization provides an isolated environment for each microservice, as each container functions independently of the others. Network connections between microservices are usually implemented through lightweight HTTP/REST or gRPC protocols, which optimize communication between services and ensure their rapid scaling.

The aim of this research is to identify the main aspects of developing CRM systems based on microservice architecture to improve the effectiveness of customer relationship management. The aim involves the fulfilment of the following research objectives:

1. Establish a connection between the introduction of microservice architecture and the flexibility and scalability of CRM systems in the business environment.
2. Evaluate the effectiveness of modern approaches to containerization and orchestration of microservices focusing on comparative analysis with traditional monolithic systems.
3. Develop recommendations for improving the development and integration of the microservice CRM architecture taking into account security issues.

2. LITERATURE REVIEW

Academic literature on the design and development of CRM systems focuses on the advantages of using microservices to ensure scalability and resilience of systems to numerous requests on servers. Researchers offer different approaches to the implementation of microservices in CRM systems, considering technical and organizational challenges. Mostofi et al. [4] emphasize that the microservice architecture enables the distribution of CRM functional components into independent services that can be developed and scaled separately. Alnofeli et al. [5] emphasize the importance of automation and continuous integration when developing CRM on a microservice architecture. The scholar [6] proves the need for automation, which reduces the probability of human errors and increases the efficiency of processes in cases where each microservice can be updated independently. According to [7], the use of containerization and orchestration are one of the key aspects of successful implementation of microservice architecture in CRM. This thesis is confirmed by [8], as it allows flexibility in the deployment and management of various services. Mena et al. [9] draw attention to the challenges associated with the integration of various microservices into a general CRM system. Tighilt et al. [10] note that it is important to ensure proper coordination between microservices, including transaction management and maintaining data consistency. On the other hand, Mazaev et al. [11] considers the security aspects of implementing microservice architecture in CRM systems.

Kazanavičius and Mažeika [12] confirms the effectiveness of the microservice approach in ensuring the adaptability of CRM systems for large companies. According to [13], the possibility of independent development and updating of individual modules significantly increases the speed of introducing new functions and reduces the risks associated with system failures. The importance of standardization of microservices development processes is emphasized in the article [14]. The author notes that the lack of a clear legal framework for regulating microservice solutions can lead to complications in their integration. Hasan et al. [15] emphasize the importance of a similar integration of DevOps methodologies in the process of developing a microservice CRM system. According to [16], automating the deployment and testing of each microservice reduces the time required for the introduction of new functions in a dynamic business environment. Hutomo and Girsang [17] emphasize the role of modern container orchestration tools in creating scalable and sustainable microservices-based CRM systems. An important work is Schröer et al. [18], which examines the security of modern digital technologies and data encryption. Gong and Cai [19] state that the use of microservices significantly increases the complexity of the security system due to the distributed nature of the architecture. This requires the implementation of additional protection mechanisms at the level of each individual service. Sildatke et al. [20] examines the economic efficiency of the transition to a microservice architecture for CRM systems. Cabrera-Gutiérrez et al. [21] note that initial costs can be significant due to the complexity of developing and implementing a new architecture. However, long-term benefits in the form of reduced support costs and the ability to quickly expand functionality make this approach promising for large companies. So, modern literature emphasizes the importance of proper design and development of microservices-based CRM systems. Technical, safety and economic issues remain poorly studied because of the constant modernization of technologies.

3. METHODS AND MATERIALS

3.1. Research design

The research procedure involves a comprehensive analysis of the use of microservice architecture for the CRM system. The first stage provided for a detailed evaluation of the main components such as customer management, sales, marketing, and support modules. It was studied how the components can be organized as separate microservices with the possibility of autonomous scaling and support of independent databases. Their interaction through the application programming interface (API) and the Representational State Transfer (REST) and Remote Procedure Calls (gRPC) protocols was evaluated to ensure effective communication between services. The second stage was a study of the implementation of microservice architecture among leading technology companies in order to analyse the reasons for such a high rate of use — about 80%. The conducted analysis demonstrated that the main reasons for choosing a microservice architecture are flexibility in development and the possibility of independent scaling. The third final stage of the research was the comparison of microservice and monolithic architecture. At this stage, recommendations are given and conclusions are drawn regarding the application of microservice architecture taking into account containerization and orchestration tools.

3.2. Sampling

Two of the most popular architectures — microservice and monolithic — are chosen for the study, which are widely used in modern CRM systems. Docker for containerization, Kubernetes for orchestration, and REST API for interaction between services were chosen for the analysis of microservice architectures. These technologies were chosen because of their popularity, high level of support from the development community, and proven performance in scalable systems. Traditional server approaches using single databases and centralized computing resources were analysed in monolithic systems.

3.3. Methods

The study employs the following methods: analytical — the use of mathematical tools when processing actual material; the logical method is used during the entire research process, including deduction, classification of material, advancement of a working hypothesis, proposing the ways of solving the problem; generalization and comparison. Research methods involved technical analysis of microservice architecture and its comparison with monolithic systems. Applied technologies were used to evaluate the possibility of microservices architecture: Docker was used to deploy and test individual microservices, and Kubernetes — to manage their scaling and maintain stable operation under high loads.

3.4. Research tools

Microsoft Excel was one of the main tools used to collect and display statistical data on the use of microservices architecture in various systems. The Python programming language was used to analyse large data volumes and automate comparisons between different architectural solutions. Docker and Kubernetes are used to simulate and test the operation of microservices in real conditions. This made it possible to evaluate their performance and flexibility at various stages of system development and support.

4. RESULTS

CRM systems consist of several key components that provide customer interaction management at various stages of business processes. The main components of CRM include modules for contact management, sales, marketing, customer service and analytics. These components allow companies to automate routine operations. The functioning of the microservice architecture assumes that each of the modules is designed as a separate unit that is scaled separately. The customer data processing module is a separate service that interacts with others through APIs. This ensures increased reliability of the system: if one of the modules fails, the others continue to work. The use of microservices allows efficient use of computing resources, distributing the load between different physical or virtual machines, which increases system performance during intensive operations.

Customer Support Service (CSS) offers a broader set of tools and services compared to the traditional model, which was reduced to phone support only. The quality of customer service functions in such a system is based on several degrees. One of them is the availability of a single database containing information about customers and contacts with them. This gives grounds to determine whether the client has previously contacted the company and for what purpose. Figure 1 shows the general components of the microservice architecture for the most popular CRM systems.

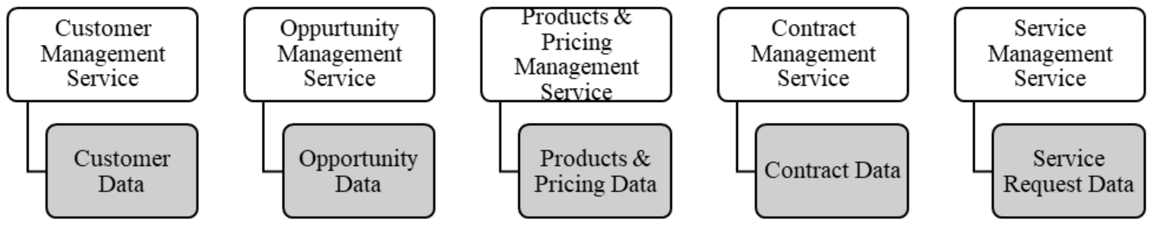
Customer Management Service
Customer Data
Oppurtunity Management Service
Opportunity Data
Products & Pricing Management Service
Products & Pricing Data
Contract Management Service
Contract Data
Service Management Service
Service Request Data

Figure 1: Microservice Architecture Of A Typical CRM Application
Source: Developed By The Authors

An important characteristic of CRM is the provision of interaction with remote customers and joint work with partners. The CSS application often monitors services recording their indicators to improve the quality of products and increase the loyalty of existing customers. Standard features of the system include a mechanism for assigning priorities, which allows you to provide a service to the client depending on the cost.

The next type of CRM systems in the classification by purpose is sales management (Sales Force Automation - SFA). He is responsible for the process of selling the company's services through various interfaces. One of the standard functions of the SFA program is to check the relevance of existing contact information, as well as to provide access to communication history across all lines. The SFA system provides management of the activities of the company’s employees (salespeople).

The functional package of the SFA application makes forecasts of further directions of work based on the data of marketing research conducted in the company. Analysis of reporting on the sales cycle identifies and analyses prospects for further development. The sales process and the results of the employees’ work in the system are analysed to add new information about sales performance to the database, as well as to track legacy factors, new marketing techniques, and trends in changing customer needs.

The use of microservices architecture has become standard among technology companies due to its numerous advantages. Netflix, Amazon, and Uber all use microservices to scale their systems. Netflix uses a microservices architecture to handle more than 200 million concurrent users by scaling each component of the system individually based on load. Amazon also applies microservices to manage its business processes, allowing each service to operate autonomously. The main reason for the introduction of microservice architecture is the ability to quickly respond to changes in the business environment. Developers can update or deploy new microservices without stopping the entire system, reducing time to implement new functions. Distribution of computing resources between microservices ensures efficient infrastructure management. Figure 2 shows the popularity of microservice architecture among large companies, which is confirmed by CodeIT data.

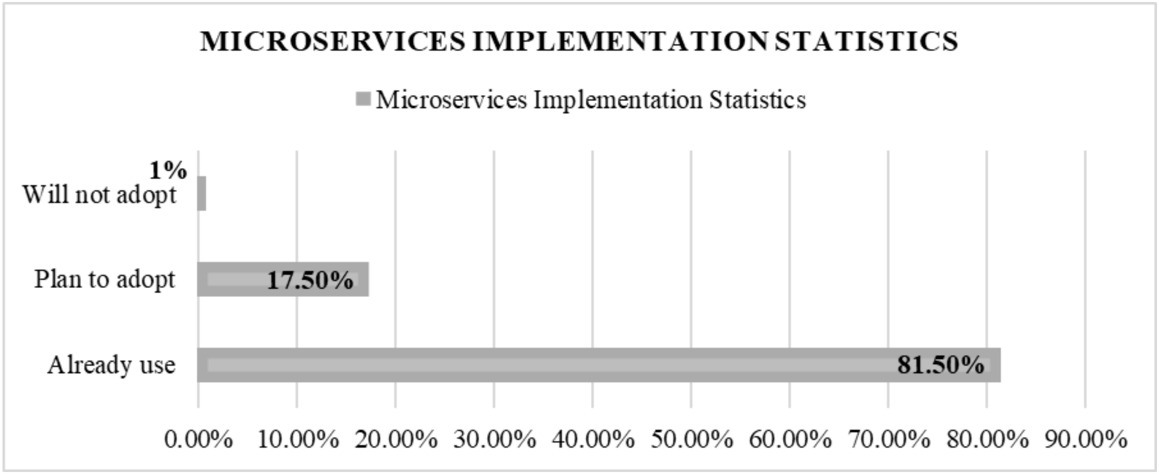

Figure 2: Microservices Implementation Statistics
Source: Developed On The Basis Of CodeIT [22]

Microservices-based applications tend to be as independent (decoupled) and centralized (bound) as possible. They contain their own domain logic and act more like filters in the classic Unix sense — they accept requests, apply logic and send a response. Instead of complex protocols such as Web Services (WS-*) or Business Process Execution Language (BPEL), they use simple REST-based protocols.

The main limitations of using microservices architecture are considered below. Increased complexity is the biggest disadvantage of microservice architecture. The complexity of a microservice programme directly correlates with the number of involved services. This type of architecture has many more moving parts than traditional software systems. Therefore, it requires additional effort, planning and automation to control communication between services, monitoring, testing, and deployment. The initiative to create a microservice-type software system will require structural changes in the team.

The two most common protocols are HTTP requests via Resource API and Simple Message Session. Teams practicing microservices architecture use the same principles and protocols as the World Wide Web (and even Unix). Frequently used resources can be cached without much effort on the part of developers or IT administrators.

Microservice architecture is used to design CRM systems where high flexibility and scalability are required. Orchestration (Kubernetes) helps to easily manage a large number of microservices, allowing the system to handle 1,000 to 5,000 requests per second, depending on the configuration. This makes the microservice architecture indispensable in cases of high intensity of work with data and the need to quickly respond to changes in business processes. This approach significantly reduces operating costs, making it possible to save up to 40% on infrastructure support thanks to the independent deployment of services.

On the contrary, monolithic architecture remains popular for smaller systems or those that do not require high flexibility and scalability. Its main advantage is ease of implementation. The functions of the system are concentrated in one application, which simplifies the process of development and security management. However, monolithic systems have significant scalability limitations, where the introduction of new functions slows down by 30%. The performance of monolithic systems significantly reduces under heavy loads, as the entire system processes requests centrally and limits the number of processed requests to 1,000 per second. Monolithic systems have higher operating costs because of the need to maintain large centralized computing capacities. They cannot be efficiently divided into separate components, as is the case with microservices. Table 1 provides a comparison between the leading systems.

Table 1: Comparison Of Microservice And Monolithic CRM Architecture

Microservice CRM architecture parameters	Indicators of microservice architecture	Indicators of monolithic architecture
Scalability	High, scaling is possible at the level of each microservice	Limited, scalability requires system-wide changes
Time of deploying	On average, it is 30% faster compared to	30% slower due to the need to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

new features	monolithic architecture	update the entire system
Operating costs	25-40% reduction due to resource ptimization	Higher operating costs due to resource intensity
Security level	Requires implementation of additional security measures at the level of each service	Fewer security requirements because all processes take place in the same system
System performance	Increases by 20% when using orchestration	Productivity is limited, reduced performance under heavy loads
The number of processed requests per second	From 1,000 to 5,000 requests depending on settings and infrastructure	Up to 1,000 requests per second on average
The number of microservices in the system	On average, 50 to 150 microservices in large systems	Usually consists of one main application

Source: Developed By The Authors

Whenever you select something, it triggers the appropriate microservices to interact via messaging systems (such as RabbitMQ), gRPC, or REST APIs to invoke your selection. Similar to monolithic modules, each microservice performs a single task that is combined to run the entire application. The problem of API management is part of the problems that arise when using a microservice architecture. This architecture is currently the leading technology for corporate application development. The issue of publishing the interface to hide the final implementation arises when designing a software product using a microservice architecture. This provides the possibility of replacing the implementation and flexibility in the distribution of roles and functions of microservices.

Three front-end technologies are mainly used in the world of CRM development - Angular, View, and React. A large CRM cannot do without big data analysis, predictive analytics and machine learning, and therefore without Python with repositories - Apache Hadoop, Spark, Scala. Cloud Native containerization is also an important issue in the field of CRM and the microservices. The containers can scale horizontally to very large sizes and serve hundreds of thousands of users without huge infrastructure costs. There is a huge variety of DevOps tools that allow the developer to do their work without worrying about how the product will be delivered. This includes both orchestrators such as Jenkins and TeamCity as well as self-testing tools such as Selenium, JMeter, and JUnit.

Security remains the main issue when using microservices architecture. Each microservice has its own individual access point, which increases the number of possible vulnerabilities in the system. This requires implementing authentication and encryption at the level of each service, monitoring traffic between microservices to detect potential threats. The lack of a single centralized point of control, which is characteristic of monolithic systems, increases the complexity of security management. Outdated technology is another challenge, as rapid development makes some tools and approaches obsolete. Traditional database management tools or server infrastructure may not be effective in distributed systems.

Therefore, microservice architecture plays a key role in modern CRM systems, ensuring their flexibility, scalability, and efficient use of resources. Microservices enable creating modular systems where each component can work independently, which increases reliability and speed of introduction of new functions. Along with the advantages, the microservice architecture creates additional security challenges that require special solutions for data protection and transaction management. The widespread use of this approach in leading technology companies demonstrates its effectiveness in large-scale business processes. However, it emphasizes the need to constantly update the technological infrastructure to support the systems' security and performance.

5. DISCUSSION

Researchers discuss integration processes and security standards in distributed systems. Mangwani et al. [23] emphasize the flexibility of the microservice architecture when using containerization, which was confirmed by his own findings. This need to use containerization is mentioned in the article [24], the increased performance of the system due to the use of Docker and Kubernetes is demonstrated. Our findings also support the conclusions of Razzaq and Ghayyur [25] on the security of microservice CRM systems, but we also found that the distributed nature of services makes it difficult to monitor security incidents in real time. According to [26], our results confirm the importance of API integration to ensure interoperability between services. However, the need to adhere to high standards of authentication and data protection is even more critical for the successful implementation of such systems. Contrary to Soldani et al. [27], our study emphasizes that coordination between microservices is more difficult in the context of large-scale CRM systems. The obtained results confirm the opinion Scatá and La Corte [28] about the importance of security measures, as the introduction of new solutions increases the complexity of system administration. Taherdoost [29] focuses on the automation of development processes, and our data indicates how automation reduces the time of introducing new features. The results of Salii et al. [30] regarding the role of Kubernetes are confirmed by our findings, however, we further note that its use requires significant resources to manage complex CRM systems. The issue of standardization is covered in an article [31], which is supported by the thesis that there are no uniform standards, which complicate the integration of new microservices. A study Pontarolli et al. [32] on the division of systems into separate microservices indicates a greater time consumption because of the burden on the project team. Therefore, the issue of introducing microservice architecture into the CRM system opens up wide prospects for further discussions. However, it requires careful coordination, safety standards, and resources to maintain the system's effectiveness. The practical use is to improve dynamic development through the distribution of functional components into independent services, which simplifies their design and implementation.

6. CONCLUSIONS

So, the article examines the technical and organizational aspects of developing microservices-based CRM systems. The research is focused on the analysis of the advantages of using microservice architecture in comparison with traditional monolithic systems. The study found that the microservice approach provides increased scalability of CRM systems, allowing each module to function independently. The roles of Docker containerization tools, which provide isolation of environments for each microservice, and orchestration (Kubernetes), which simplifies the management of complex systems, are outlined. Security issues are considered, where the microservice architecture, because of its distributed nature, requires the implementation of additional measures to protect data and ensure consistency between services. Despite the initial implementation costs, microservices-based CRM systems reduce operating costs in the long run due to the ability to scale independently. The research results give grounds to develop recommendations for the use of modern automation and monitoring tools, which will contribute to the effectiveness of microservices-based CRM system support. A CRM system has been developed, which differs from analogues in its versatility, in order to manage the company's processes, because it can be used by both ordinary employees and senior managers.

The general requirements for the CRM system are formulated, namely security, reliability and availability, clear interface, ease of implementation and support, availability of mandatory basic functions. Complex operating systems have specific functions and delivery of the system in the form of a web application. Current technologies for building web applications were studied and technologies used in the work were described, including Django, Django Rest Framework, React, MobX, Ant Design. The technologies were successfully applied during the implementation of the system. The aim of the research was achieved taking into account the general and specific requirements of the relevant business processes in the companies. Further research should focus on the possibility of scaling with new technologies and improving the quality of processing requests and server data.

REFERENCES:

[1] M. S. Ferreira, J. Antão, R. Pereira, I. S. Bianchi, N. Tovma, and N. Shurenov, "Improving real estate CRM user experience and satisfaction: A user-centered design approach", *Journal of Open Innovation: Technology, Market, and Complexity*, Vol. 9, No. 2, 2023, art. 100076. doi: 10.1016/j.joitmc.2023.100076.

[2] L. Wang, P. Hu, X. Kong, W. Ouyang, B. Li, H. Xu, and T. Shao, "Microservice architecture recovery based on intra-service and inter-service features", *Journal of Systems and Software*, Vol. 204, 2023, art. 111754. doi: 10.1016/j.jss.2023.111754.

[3] Ł. Szwałek, and J. Smołka, "Choosing the optimal database system to create a CRM system", *Journal of Computer Sciences Institute*, Vol. 26, 2023, pp. 48-53. doi: 10.35784/jcsi.3079.

[4] V. M. Mostofi, E. Krul, D. Krishnamurthy, M. Arlitt, "Trace-driven scaling of microservice applications", *IEEE Access,* Vol. 11, 2023, pp. 29360-29379. doi: 10.1109/ACCESS.2023.3260069.

[5] K. Alnofeli, S. Akter, and V. Yanamandram, *Understanding the future trends and*

innovations of ai-based crm systems, in: S. Akter, S.F. Wamba (Eds.), Handbook of big data research methods, Edward Elgar Publishing Ltd, Northampton, 2023, pp. 279-294. doi: 10.4337/9781800888555.00021.

[6] R. Ouyang, J. Wang, H. Xu, S. Chen, X. Xiong, A. Tolba, and X. Zhang, "A microservice and serverless architecture for secure IoT system", *Sensors*, Vol. 23, No. 10, 2023, art. 4868. doi: 10.3390/s23104868.

[7] C. Ramonell, R. Chacón, and H. Posada, "Knowledge graph-based data integration system for digital twins of built assets", *Automation in Construction*, Vol. 156, 2023, art. 105109. doi: 10.1016/j.autcon.2023.105109.

[8] I. G .R. Wijaya, and A. N. Fajar, "A design study of microservice architecture on white label travel platform", *Journal of System and Management Sciences,* Vol. 13, No. 4, 2023, pp. 249-264. doi: 10.33168/JSMS.2023.0415.

[9] M. Mena, J. Criado, L. Iribarne, A. Corral, R. Chbeir, and Y. Manolopoulos, "Towards high-availability cyber-physical systems using a microservice architecture", *Computing*, Vol. 105, No. 8, 2023, pp. 1745-1768. doi: 10.1007/s00607-023-01165-x.

[10] R. Tighilt, M. Abdellatif, I. Trabelsi, L. Madern, N. Moha, and Y. G. Guéhéneuc, "On the maintenance support for microservice-based systems through the specification and the detection of microservice antipatterns", *Journal of Systems and Software*, Vol. 204, 2023, art. 111775. doi: 10.1016/j.jss.2023.111755.

[11] G. Mazaev, M. Weyns, P. Moens, P. J. Haest, F. Vancoillie, G. Vaes, J. Debaenst, A. Waroux, K. Marlein, F. Ongenae, and S. Van Hoecke, "A microservice architecture for leak localization in water distribution networks using hybrid AI", *Journal of Hydroinformatics,* Vol. 25, No. 3, 2023, pp. 851-866. doi: 10.2166/hydro.2023.147.

[12] J. Kazanavičius, and D. Mažeika, "An approach to migrate from legacy monolithic application into microservice architecture", *2023 IEEE Open Conference of Electrical, Electronic and Information Sciences, eStream 2023 – Proceedings*, Vilnius, Lithuania, April 27, 2023. doi: 10.1109/eStream59056.2023.10135021.

[13] A. Bojanowska, and M. Kulisz, "Using fuzzy logic to make decisions based on data from customer relationship management systems", *Advances in Science and Technology Research Journal*, Vol. 17, No. 5, 2023, pp. 269-279. doi: 10.12913/22998624/172374.

[14] S. Robitzsch, M. Centenaro, N. di Pietro, L. Cordeiro, A. S. Gomes, P. Sanders, and A. Ishaq, "Prospects on the adoption of a microservice-based architecture in 5G systems and beyond", *Computer Networks*, Vol. 237, 2023, art. 110058. doi: 10.1016/j.comnet.2023.110058.

[15] M. H. Hasan, M. H. Osman, N. I. Admodisastro, and M. S. Muhammad, "From monolith to microservice: Measuring architecture maintainability", *International Journal of Advanced Computer Science and Applications*, Vol. 14, No. 5, 2023, pp. 857-866. doi: 10.14569/IJACSA.2023.0140591.

[16] A. V. Tokmak, A. Akbulut, and C. Catal, "Boosting the visibility of services in microservice architecture", *Cluster Computing*, Vol. 27, No. 3, 2024, pp. 3099-3111. doi: 10.1007/s10586-023-04132-5.

[17] H. W. Hutomo, and A. S. Girsang, "Implementations of microservice on self-service application using service oriented modelling and architecture: A case study", *Journal of System and Management Sciences*, Vol. 13, No. 3, 2023, pp. 205-218. doi: 10.33168/JSMS.2023.0314.

[18] C. Schröer, S. Wittfoth, and J. M. Gómez, *Multi-metric approach for decomposition of microservice-based data science workflows*", in: T. Batista, T. Bures, C. Raibulet, H. Muccini (Eds.), Software Architecture. ECSA 2022 Track and Workshops, Springer Science and Business Media Deutschland GmbH, Cham, 2023, pp. 355-369. doi: 10.1007/978-3-031-36889-9_24.

[19] J. Gong, and L. Cai, "Analysis for microservice architecture application quality model and testing method", *2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Snpd-Winter 2023,* Taiyuan, China, July 5-7, 2023, pp. 141-145. doi: 10.1109/SNPD-Winter57765.2023.10223960.

[20] M. Sildatke, H. Karwanni, B. Kraft, and A. Zündorf, "A distributed microservice architecture pattern for the automated generation of information extraction pipelines", *SN Computer Science*, Vol. 4, No. 6, 2023, art. 833. doi: 10.1007/s42979-023-02256-4.

[21] A. J. Cabrera-Gutiérrez, E. Castillo, A. Escobar-Molero, J. Cruz-Cozar, D. P. Morales, and L. Parrilla, "Blockchain-based services implemented in a microservices architecture

using a trusted platform module applied to electric vehicle charging stations", *Energies*, Vol. 16, No. 11, 2023, art. 4285. doi: 10.3390/en16114285.

[22] CodeIT, "Benefits of microservices architecture", 2022. Available in: https://codeit.us (21.12.24).

[23] P. Mangwani, N. Mangwani, and S. Motwani, "Evaluation of a multitenant SaaS using monolithic and microservice architectures", *SN Computer Science,* Vol. 4, 2023, art. 185. doi: 10.1007/s42979-022-01610-2.

[24] G. Filippone, N. Qaisar Mehmood, M. Autili, F. Rossi, and M. Tivoli, "From monolithic to microservice architecture: An automated approach based on graph clustering and combinatorial optimization", *Proceedings - IEEE 20th International Conference on Software Architecture, ICSA 2023,* L'Aquila, Italy, March 13-17, 2023, pp. 47-57. doi: 10.1109/ICSA56044.2023.00013.

[25] A. Razzaq, S. A. K. Ghayyur, "A systematic mapping study: The new age of software architecture from monolithic to microservice architecture-awareness and challenges", *Computer Applications in Engineering Education*, Vol. 31, No. 2, 2023, pp. 421-451. doi: 10.1002/cae.22586.

[26] O. M. Al-Shuridah, and N. O. Ndubisi, "The effect of sustainability orientation on CRM adoption", *Sustainability*, Vol. 15, No. 13, 2023, art. 10054. doi: 10.3390/su151310054.

[27] J. Soldani, J. Khalili, and A. Brogi, "Offline mining of microservice-based architectures (extended version)", *SN Computer Science*, Vol. 4, 2023, art. 304. doi: 10.1007/s42979-023-01721-4.

[28] M. Scatá, and A. La Corte, "A complex insight for quality of service based on spreading dynamics and multilayer networks in a 6G scenario", *Mathematics*, Vol. 11, No. 2, 2023, art. 423. doi: 10.3390/math11020423.

[29] H. Taherdoost, *Customer relationship management*, in: EAI/Springer Innovations in Communication and Computing, Springer Science and Business Media Deutschland GmbH, Cham, 2023, pp. 237-264. doi: 10.1007/978-3-031-39626-7_10.

[30] S. Salii, J. Ajdari, and X. Zenuni, "Migrating to a microservice architecture: Benefits and challenges", *2023 46th ICT and Electronics Convention, MIPRO 2023 – Proceedings,* Opatija, Croatia, May 22-26, 2023, pp. 1670-1677. doi: 10.23919/MIPRO57284.2023.10159894.

[31] S. Pallewatta, V. Kostakos, and R. Buyya, "Placement of microservices-based IoT applications in fog computing: A taxonomy and future directions", *ACM Computing Surveys*, Vol. 55, No. 14S, 2023, art. 321. doi: 10.1145/3592598.

[32] R. P. Pontarolli, J. A. Bigheti, L. B. R. de Sá, and E. P. Godoy, "Microservice-oriented architecture for industry 4.0", *Eng*, Vol. 4, No. 2, 2023, pp. 1179-1197. doi: 10.3390/eng4020069.