УДК 004.27:004.4
DOI: 10.31673/2409-7292.2024.030009

Koretskyi O.

ARCHITECTURE OF MULTIFUNCTIONAL APPLICATION SOFTWARE MICROSERVICES

One of the key challenges in modern software architecture is ensuring the efficient management of resources and computational power in distributed systems. As the demand for real-time information services grows, traditional architectures face limitations in handling dynamic workloads and maintaining low latency. The structure of the micro-service architecture of a multifunctional software application is presented. The proposed structure advances existing architectures by introducing a Broker block, which creates conditions for implementing models and methods of dynamic foggy calculations. These calculations aim to improve the quality of information services by migrating necessary software microservices to regions with high demand. This migration enhances efficiency in resource utilization and service delivery. The example provided demonstrates that distributed dynamic foggy calculations can significantly boost the computing power of information networks by offloading some of the network traffic, reducing congestion. This also minimizes the round-trip delay, optimizing overall performance. Additionally, by dynamically distributing computing tasks based on current demand, the system improves both scalability and responsiveness. The implementation of such foggy computational techniques offers a solution to the growing need for real-time data processing in modern applications, particularly in environments with fluctuating network traffic and computational demands. This approach is especially relevant for services requiring low latency and high availability, making it a powerful tool in the development of future network architectures.

Keywords: micro -service architecture, software, dinamic foggy.

Introduction

Software for various services, in modern times, is a fairly large part of existing and future information systems, most of which still “store” quite old software architecture and methods, which essentially represent a kind of software “monolith”.

From the point of view of analyzing the concept of software architecture, it is worth paying attention to the very concept of the word “Architecture” in relation to information and telecommunication systems. Based on the concept of an information system, its architecture from a constructive point of view is a set of key decisions that remain unchanged when business technology changes within the framework of the business vision (concept) [1, 2]. This leads to the following conclusion: if there is a need to change key decisions in an IT product when changing business technology (business model/processes/products) within the business vision (concept), then the cost of system ownership increases sharply. As a result, there is an understanding of the importance and necessity of adopting the system architecture as a company standard due to the significance of architectural solutions and their stability in relation to changes in business technologies. It is important to note that the issue of system architecture should be addressed at the level of conceptual design, and in some cases at the stage of feasibility study of the system [1, 6].

Any service provided on the network in fact is Software (usually high-level) - the product of the beneficiary company. Accordingly, the same questions can be asked not only to the IT system as a whole, but also to the Software that implements either an entire IT product or its component part [1, 6].

In modern times, in the world of software development, a sufficient amount of knowledge has been obtained in the field of developing effective code, methods, principles and approaches to the implementation of an IT product (from a code point of view) [2–4]. There are also various architectural solutions. These include: monolithic, multi-level, event-driven, serverless and microservice [1, 6].

For example, the serverless architectural style is applicable to applications that use other services (third parties) as a solution in order to solve the problem of backend and server load. Serverless architecture can help reduce time spent working on regular server tasks. One famous example of a serverless API is Amazon's AWS "Lambda" service. We can also include the so-called serverlet programming in this architectural style [1, 6].

© Koretskyi, O. (2024). Architecture of multifunctional application software microservices. Сучасний захист інформації, 3(59), 85–93. https://doi.org/10.31673/2409-7292.2024.030009.

However, especially in the last 3-4 years, the microservice architectural style of software development and implementation has been actively developing. Especially after its successful pilot application in such well-known IT products as Amazon and Netflix. This architecture is important for infrastructure solutions, such as Internet of Things platforms, Web platforms that implement more than one product, and others.

It is worth noting that recently not only new products have been developed in this architectural style, but also some companies are literally rewriting existing products, or trying to find the level of decomposition of former monoliths in order to maintain a balance between the cost of development and the transition to a new architecture while maintaining all functions. opportunities, and receiving the benefits that microservice software architecture provides.

The software is a set of software blogs - microservices and the logic of their interaction. At the same time, a microservice can be presented in the form of an entire subsystem that implements a set of service functions, or in the form of a fairly atomic software structure - a function. The level of “atomicity” and independence of microservices is determined by the development team in each specific project, including taking into account software development tools. However, ideally, microservices should be independent, and if it is necessary to scale system functions, certain microservices should be multiplied to meet increased requirements. At the same time, when cloning microservices, they must automatically connect to the general “microservice organism” of the information system so that during further processes these microservices perform their functions without errors, interact with other microservices, with client software, and so on. It is also worth noting that the so-called cloning/reproduction of microservices is ensured, among other things, by the principles of software virtualization using various tools, including containerization. Live migration is provided using computing system orchestration tools [6, 13].

Thus, the above-mentioned synergies of infrastructure technologies can be harmoniously complemented by the development architecture and deployment principles of microservice software services. Especially in this topic, there is interest in combining fog computing solutions and microservices. This combination will allow us to achieve a sufficiently “dense fog” with proper atomization of service microservices [8, 13].

A number of works are devoted to the development of a microservice software architecture structure that allows for the implementation of models and methods of dynamic fog computing aimed at improving the quality of presentation of information services through the migration of the necessary software microservices to an area with an increased demand for information [5–11].

Literature review

A number of works are devoted to the development of a microservice software architecture structure that allows for the implementation of models and methods of dynamic fog computing aimed at improving the quality of presentation of information services through the migration of the necessary software microservices to an area with an increased demand for information [1, 2, 5, 6, 8–11].

General issues of substantiation of the principles of development and use of microservice software architecture, which is intended to ensure the implementation of models and methods of dynamic cloud computing, aimed at improving the quality of presentation of information services, were considered in works [1, 2, 6, 8]. The material presented in these works reveals the general idea of implementing microservice software architecture, the principles and methods of its development and use in order to improve the quality of providing information support services in global information networks. Solving the partial task of developing a microservice architecture under the condition of ensuring the implementation of models and methods of dynamic fog computing, aimed at improving the quality of the presentation of information services, was not considered in these works.

Scientific work [5] is devoted to solving the issue of defining a preliminary structure and analysis in the form of a taxonomy of construction concepts, which covers the entire life cycle of the functioning of microservices, as well as the organizational aspects of building their architecture.

Direct issues related to solving the problem of ensuring the implementation of models and methods of dynamic fog computing by constructing a convenient software microservice structure are not considered in this work.

Article [10] presents a model of a formal structure for determining and testing the effectiveness and quality of the implementation of information support process flows based on microservices. The basis of the formal semantics of the presented structure in the article is determined by Petri Net, which supports the determination of time constraints. In particular, the structure presented is aimed, first of all, at solving the problem of determining and verifying the communication behavior of microservice systems among themselves and with external services.

The issue of improving the quality of information provision by applying an improved model of microservice software architecture, which ensures the implementation of fog computing models, was not resolved in the work.

Issues of developing and assessing the effectiveness of a microservice software architecture designed to solve one of the applied problems of information support are discussed in article [11]. This work examines the issues in sufficient detail and presents the results of solving the problem of developing one of the options for microservice software architecture aimed at implementing a particular task of information support. Conceptual approaches to ensuring the implementation of models and methods of dynamic fog computing through the use of the given version of microservice software architecture were not considered in this work.

The research materials published in the article [9] solve the scientific task of transforming the monolithic software structures created at the previous stages into microservice systems of promising software. This article contributes to shed light on migration approaches, microservice identification techniques, factors that drive migration, tools used during the migration, migration issues and benefits. At the same time, the requirements for the construction of the microservice architecture, which will ensure the effectiveness of the implementation of fog computing, are not considered in this work.

An analysis of the above publications showed that at this time, no attention has been paid to the development of a new holistic structure of microservice architecture, style of development and implementation of software applications capable of ensuring the implementation of dynamic fog computing and the migration of the necessary software microservices to an area with increased demand for the provision of information provision services due attention.

These circumstances determine the relevance and form a new scientific problem, the solution of which this work is devoted to.

Research Methodology

The purpose of this article is to develop and substantiate the structure of a microservice architecture of a multifunctional software application that can provide implementation of models and methods of dynamic fog computing aimed at improving the quality of presentation of information services.

The basic architectural approach to building service software is usually defined as a “set of answers” to the following questions [6, 13]:

What purpose is the system designed to serve?

What are the target functions of the system?

What parts is it divided into?

How do these parts interact?

Where are these parts located (virtually/physically)?

What basic principles of operation does the system implement?

Thus, the system architecture is a logical structure, a model, and fully affects the total cost of ownership or, if an IT product, then the cost of modification/change/improvement, through a set of related decisions on the choice of implementation tools, operating platform, database management systems, databases, telecommunications, architecture of the software itself.

At the moment, given the comprehensive growth of various kinds of applications, platforms (software products on the Internet) and their demand from users, new software development architectures are changing and proposed. The main reasons for the change and the very need for fundamentally new and effective software architectures lie in the need for the product to be able to change and provide new functions for users in a short time. In modern times, the paradigm for creating final products (for the user) has changed, which is primarily based on an analysis of the user's needs. And if the product is not user-friendly, then it will have no place in the market. In this vein, when changing any business logic, software modifications should be minimal and not affect existing (built-in) processes in the system.

In addition, given the complexity (in the engineering sense of the word) of software solutions, there are the following requirements for software of modern and promising services [2, 4, 5]:

flexibility in configuration;

fast scalability (both individual functions and the entire system);

sustainability;

portability (that is, independence from the deployment and maintenance environment);

safety;

speed of software development;

the ability to work with several independent teams on software development/testing;

modularity.

Based on the analysis of modern architectural approaches to software development, one of the most actively developing architectural approaches to the development of high-load applications is microservice architecture [1, 8]. This architecture is designed to solve the problem of creating complex systems by decomposing them into independent modules. The MVC decomposition level - Model-View-Controller, server modules, classes, objects, databases, or even individual function-methods) is determined in each particular case, project by the development team. However, each of the modules defined above can be represented as conventional interpreted objects, such as a container or a virtual machine.

It should be noted that the complex architecture of generalized software, built on the principle of micro-service architecture, requires solving a number of problems of implementing asynchronous interaction between system modules, discovering new microservices in the system, as well as implementing queues for receiving requests and responses to received requests [1, 8] . Accordingly, solving such problems requires the presence of a separate functional block in the microservice architecture.

The task of which is to create conditions for the implementation of models and methods of dynamic fog computing aimed at improving the quality of presentation of information services through the migration of the necessary software microservices to an area with increased demand for information [6, 13].

Provided that the decomposition is not low-level, it seems appropriate to present the architecture of software that is developed according to the microservice approach in the form of a specific structural diagram, built in the form of related key functional blocks.

We present the key elements of a typical microservice architecture of server software in the form of a functional diagram shown in Fig. 1.

Here are brief descriptions of some subsystems and their main functions:

Balancer HAProxy. This is a software model (performs the functions of a proxy server) that accepts user requests and performs the functions of TCP/HTTP load balancing, as well as distributing requests between subsystems (for example, software modules) on the backend software. In this case, requests can be generated by various user devices (mobile application, smart device, browser, and so on).

Note: the “Balancer HAProxy” module and WEB server in practice are usually implemented (for external programmers) in the form of ready-made (single) WEB server software on which the solution is developed and will be deployed. For example, Nginx server.

API GW (Application Programming interface Gateway). This software module can be a separate microservice (or it can be a WEB server software module, for example, Nginx Gateway), which acts as a single point for receiving requests from external users of the microservices API. The GW API receives a request and, based on the URL, determines which microservice should receive and process it.

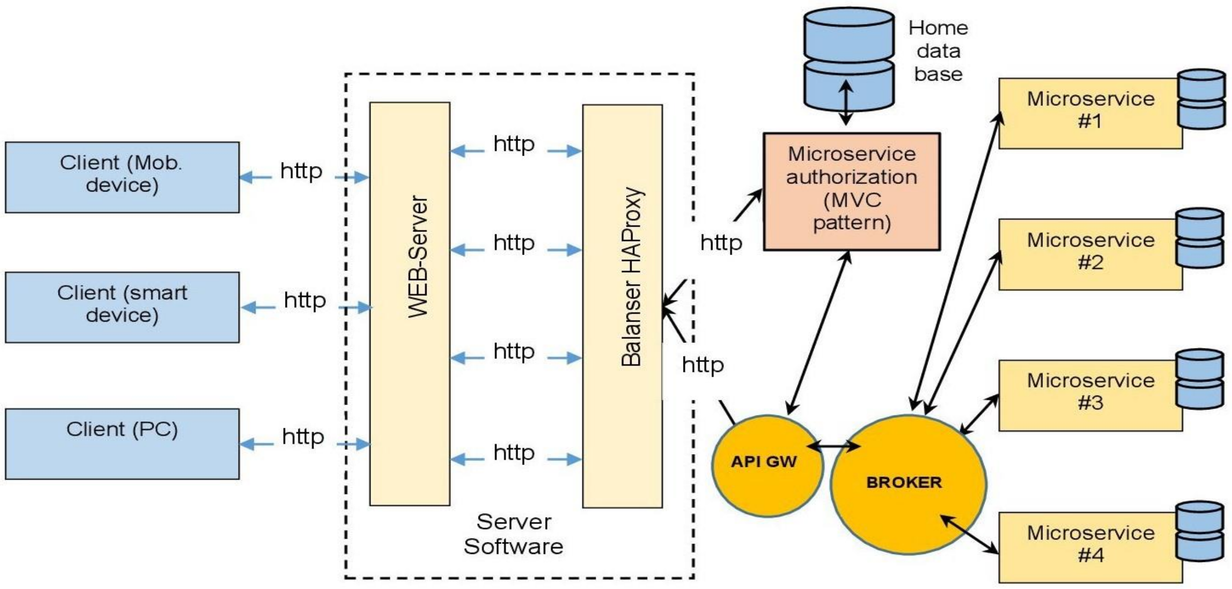

Fig. 1. Typical microservice architecture high level software

After this, the recipient microservice transmits the message through the "Broker" module, waits for a response and returns it back. The second important function of the GW API is the creation and maintenance of a user session. To create a user session, the Token tool is usually used, which is generated for each new user and transmitted in the header of internal packets (or the body of the message in a special JSON or XML field). The token allows you to track all session data.

MAIN DB – the main Database of the application/platform. This database stores information about users and related information, as well as other application metadata.

Brocker – message broker. This module is primarily necessary to implement asynchronous interaction between system modules. In particular, it allows you to implement the functions of a service for detecting new microservices in the system. One of the main functions of the module is the implementation of queues for receiving requests and responses to received requests.

Results

The given typical high-level microservice software architecture is implemented as follows.

Each of the microservices connects to this message broker using the corresponding queues. The broker also performs the functions of routing messages based on the metadata transmitted in the message headers (including data about the user session, as previously mentioned - the authorization token). Thus, each instance of a microservice of a certain type subscribes to listen to the request queue and performs message collection.

This process is similar to the distribution of monolithic software applications into microservices with all the ensuing private tasks and the resulting difficulties of subsequent implementation and operation without losing the quality of management of the microservice system [4, 7].

In this case, it is worth noting that when sending requests, the recipient is not a specific microservice instance, but the type to which it belongs. Since, within the framework of the microservice approach, the implementation of each of the microservices can be done in such a way that to increase the performance of the microservice, only its copying is required. Accordingly, live migration of individual microservices, or types of microservices, and their joint work at a geographical distance from each other is possible. The data broker also takes on the load balancing function. In this case, balancing is performed implicitly by the microservices themselves servicing the request queue [4, 7].

In addition, it is worth noting that existing solutions in the form of open source brokers provide the implementation of additional functions that make it possible to monitor data flows passing between microservices.

The microservices architectural approach is implemented differently in each development case, and the broker is not a required element. However, in a software solution where discrete tasks are resolved, building an architecture taking into account the broker allows us to obtain a turnkey solution to some of the problems of interaction between microservices. For example, the same function is the asynchronous interaction of software modules or the discovery of new microservices.

In general, the presence of a "Broker" that solves certain problems aimed at ensuring the conditions for the implementation of models and methods of dynamic fog computing aimed at improving the quality of the provision of information services through the migration of the necessary software microservices to an area with an increased demand for information.

It is also worth noting that when developing small systems, the tasks of a broker, software gateway, "balancer" can be resolved by additional modules of the main WEB server on which the software product is developed and deployed; for example, the latest version of Nginx provides such capabilities.

However, for disparate, complex, high-performance platforms and solutions, their own architecture is built, often combining other systems. With all the variety of architectural approaches to the implementation of server software, it is worth noting that individual independent software modules can be implemented in their own "isolated" environment and presented in the form of virtual machines or containers. This opens up the possibility of using less powerful devices as a "computing cloud," namely Fog devices. At the same time, the less demanding the microservice is in terms of computing power (for example, the level of software decomposition at the level of lambda functions), the less productive the Fog Computing device can be used - up to Smart devices or nano-computers (if we talk about the concept of Nano-networks).

It should be noted that, conceptually, micro -service architecture generally forms a sufficiently branched data network. The task of which is to provide a variety of data traffic. At the same time, data transmission involves the main units, international data stations, channels that connect them and national processing and transmission units.

The use of such data transmission network, built on the principles of micro -service architecture, forms another problematic issue and requires the search for ways to solve it. Namely, working management of the entire multi -element system, taking into account the requirements and provisions of the theory of data transmission.

The principle of managing any network of communication is to interact with key blocks of management of support of the readiness of the communication network, the formation of the necessary management processes and management of them in the course of completing the tasks set [4, 7].

One of the aspects of achieving the goal of management is to ensure the use of all communication networks in solving the tasks set. The achievement of this goal is related to the solution of a number of tasks, the main of which are: collection, processing and analysis of data of the network; preparation and decision -making to eliminate failures and faults; development of automated controls on the network; bringing tasks to performers; organizing and maintaining the interaction of subsystems; management organization during tasks; Reservation and control of subsystems.

The main indicator of the quality of functioning of the transport communication network (TCN) is the promptness of management decisions on the state of TCN in the conditions of high dynamics of change of states of TCN [4, 7]. Among the set of various factors that can affect the specified promptness, we can distinguish a number of factors related to the need to form a large set of solutions to change the configurations of individual routes, fragments and all network, depending on the condition a process related to the possibility of using artificial intelligence elements.

Currently, in the development of new principles and methods of managing complex processes in branched data systems of data transmission systems have been widely used multagent technologies - technologies for the development and use of multi -gentagent systems - complex systems operating with the help of several intellectual agents, the process of self -organization of which is reduced to consistency, interactions of agents among themselves

In addition to the solution of the task of choosing the optimally, the order of control of the configuration of the transmission routes in the multi -service system, in addition to the solution to the optimal way to choose the length of the data cycle formed by the system formed by the system. And also, through the application of a hierarchical approach to the construction of a system of management with the use of multi-gapontic technologies to manage the state of the network at the technological and higher operational and technical level [4, 7].

It is obvious that the use of multi -agent technologies in the development of the micro -service management system will significantly increase the efficiency of its functioning and directly increase the prompt decision -making indicators of data management in data management in the conditions of rapid dynamics of changes in the condition of individual network elements of TMZ.

The proposed conceptual model of the network management system should have a number of distinctive features and capabilities. Namely:

availability of information links between the components of the micro-service network;

the transition of the system to a new state when changing external influences;

the ability to determine management goals;

availability of a multi -gaigent control system, which allows to form a set of decisions in the presence of configuration controls for changes in the configuration of individual routes, fragments and the entire TCN and the choice of the most optimal is consistent with the decision -making person;

the ability to choose an acceptable process of control and management of the state of the object of the object of management;

preliminary assessment of the duration of the management cycle, which is formed by the multi -agent management system and is consistent with the decision -making person.

The distribution of the multi-gaignt system by different levels of construction of the network management system allows you to control and control the condition of individual elements at different levels.

It is obvious that the management of a micro -network network is a component of the total array of tasks that rely on a "broker"

It is obvious that the management of a micro -network network is a component of the total array of tasks that rely on a "broker"

As part of the research of this article, it is urgent to evaluate the amount of the duration of the management of the condition of the micro -service network. It is convenient to use consistent assessment algorithms. The most suitable of these types of algorithms in terms of computational complexity, accuracy and speed of the obtained results is Robbinson-Montro [10, 11].

Let us give an example of using the proposed structure, namely a framework for the interaction of fog and edge computing in the form of a functional diagram of a service for processing user photos using artificial neural networks in order to improve the quality of pictures taken on user devices.

A similar service is provided within the Google Photos application. After the user has taken a picture with the camera of a mobile device, this picture is sent to the nearest microservice, which sends the picture for preparation for processing by an artificial neural network (for example, a convolutional type) [3, 12].

Next, the processed image is processed in a microservice where the ANN is implemented. This example implements a variant of a trivial service that can be deployed in this framework. For example, one of the most interesting and relevant areas, namely video/photo content processing. It is implemented on the basis of an augmented reality application (AR - Augmented Reality) or a virtual reality application (VR - Virtual Reality). These types of applications require high-quality, high-definition content.

As a result of the implementation of the above requirements, additional tasks arise to provide the necessary network resources and computing capabilities for their processing. For example, layering augmented content into reality. In such applications, the first step is to determine the token to which the virtual content is bound. Such a marker can be the surrounding environment, for example, the entrance to a museum, and so on. At the same time, to recognize objects in a photograph, for better quality, machine vision solutions based on artificial neural networks are required. As a result, it can be determined that for the widespread introduction of AR/VR applications into people's lives and, possibly, into every type of human activity, fairly large network and computing resources are required. And distributed fog dynamic computing can help implement this direction, providing enormous computing capabilities to the surrounding world (the sum of surrounding devices) and reducing the load on the network component of communication networks and at the same time minimizing the network component of the application's round-trip delay.

Conclusions

The work presents the structure of a microservice software architecture for a richly functional software application.

1. The submitted structure, in the development of existing architectures, will include the "Broker" block, the task of which is to provide conditions aimed at ensuring the implementation of models and methods of dynamic fog computing aimed at improving the quality of the provision of information services through the migration of the necessary software microservices to an area of high demand for information.

2. The tasks of the "Broker" are defined, namely:

implementation of asynchronous interaction between system modules;

implementation of the service function for detecting new microservices in the system;

implementation of queues for receiving requests and responses to received requests;

message routing function based on metadata transmitted in message headers.

3. Let us give an example of using the proposed structure of microservice software architecture. Namely, a framework for the interaction of fog and edge computing, presented in the form of a functional diagram of the service using the example of improving the quality of photographic images taken using artificial neural networks.

The given example showed that distributed fog dynamic computing can provide enormous computing capabilities to the information network (the sum of surrounding devices) by reducing the load on the network component of communication networks and at the same time minimizing the network component of the application's round-trip delay.

4. Presented microservice architecture with the inclusion of the "Broker" block offers a robust solution for enhancing the efficiency of information service provision in distributed systems. The dynamic fog computing model not only optimizes the migration of microservices to high-demand areas but also effectively manages system resources and network load. This approach ensures better scalability, flexibility, and adaptability in various computational environments, especially in systems where low latency and high availability are crucial. Moreover, the proposed architecture fosters seamless communication between system modules and ensures effective message routing, contributing to improved service performance. Future research and development can focus on refining the integration of artificial intelligence techniques for further optimizing microservice management, particularly in edge and fog computing frameworks, to continue enhancing the computational power of such systems while minimizing delays and maximizing resource utilization.

References

1. Baresi, Luciano, Garriga Martin, “Microservices: The Evolution and Extinction of Web Services” In book: Microservices, Science and Engineering, 2019, pp.3-28, doi:10.1007/978-3-030-31646-4_1.
2. S. K. Boell, D. Cecez-Kecmanovic.: What is an Information System? 48th Hawaii International Conference on System Sciences – HICSS. Kauai, HI, USA, 2015, pp. 4959-4968, doi: 10.1109/HICSS.2015.587.
3. M. Bukovčan, D. Blažević, K. Nenadić and M. Stević.: Clean Architecture of Client-Side Software Development for Smart Furniture Control. 11th Mediterranean Conference on Embedded Computing – MECO. Budva, Montenegro, 2022, pp. 1-4, doi: 10.1109/MECO55406.2022.9797122.
4. Fei Chen, Wei Ren.: On the Control of Multi-Agent Systems: A Survey. Foundations and Trends. Systems and Control 6(4), 2019, 339-499. http://dx.doi.org/10.1561/2600000019. 2019.
5. Garriga M.: “Towards a Taxonomy of Microservices Architectures” In Software Engineering and Formal Methods. Springer, Lecture Notes in Computer Science 10729, 2018, 203-218. https://doi.org/10.1007/978-3-319-74781-1_15
6. Gabbrielli, M., Giallorenzo, S., Guidi, C., Mauro, J., Montesi, F.: Self-reconfiguring microservices. In: Theory and Practice of Formal Methods. LNCS 9660, pp. 194–210. Springer, Cham 2016. https://doi.org/10.1007/978-3-319-30734-3_14
7. Gharbi A., Gharsellaoui H., Ben Ahmed S.: Multi-Agent control system. 9th International Conference on Software Engineering and Applications –ICSOFT-EA. 2014. pp. 117-124. https://doi.org/10.5220/0005001101170124
8. Isha V P, Vishalakshmi Prabhu H.: An Approach to Clean Architecture for Microservices Using Python. 7th International Conference on Computation System and Information Technology for Sustainable Solutions – CSITSS, Bangalore, India, 2023, pp. 1-5/ doi: 10.1109/CSITSS60515.2023.10334229.
9. Martínez Saucedo. et al.: Migration of Monolithic Systems to Microservices. 2023, 1-37. Available at SSRN: https://ssrn.com/abstract=4584794, http://dx.doi.org/10.2139/ssrn.4584794
10. S. Speth, S. Stieß, S. Becker.: A Saga Pattern Microservice Reference Architecture for an Elastic SLO Violation Analysis. IEEE 19th International Conference on Software Architecture Companion – ICSA-C, Honolulu, HI, USA, 2022, pp. 116-119. doi: 10.1109/ICSA-C54293.2022.00029.
11. Strand R.D., Kristensen L.M., Petrucci L.: Development and Verification of a Microservice Architecture for a Fire Risk Notification System. Lecture Notes in Computer Science 14150 LNCS, 2023, pp. 27 – 53. doi: 10.1007/978-3-662-68191-6_2
12. Y. Romani, O. Tibermacine, C. Tibermacine.: Towards Migrating Legacy Software Systems to Microservice-based Architectures: a Data-Centric Process for Microservice Identification. IEEE 19th International Conference on Software Architecture Companion – ICSA-C, Honolulu, HI, USA, 2022, pp. 15-19. doi: 10.1109/ICSA-C54293.2022.00010.
13. Zhiding Li, Chenqi Shang, Jianjie Wu, Yuan Li.: Microservice extraction based on knowledge graph from monolithic applications. Information and Software Technology 150, 2022, 106992, https://doi.org/10.1016/j.infsof.2022.106992.

Надійшла 30.07.2024