DOI 10.34185/1562-9945-6-125-2019-12
УДК 004.046

L. Bozhukha, O. Biloborodko

SELECTING THE STRATEGY FOR DESIGNING THE SOFTWARE ARCHITECTURE

Annotation. The article presents the results of the analysis of the existing web application architecture design methodologies and the corresponding software development tools and resources. The shortcomings of the existing approaches to application development were identified and conducted the analysis can be used to design the web systems architecture. The practical value of the work is the development of algorithms for creating, deploying and scaling applications using different software. A test system based on several classic approaches has been developed.
Keywords: web service, microservice architecture, containerization, SOA, EDA.

Formulation of the problem. Due to the increasing complexity of business processes, the amount of data in computing systems, the increasing requirements for modern software systems, the process of designing, developing and operating software requires significantly greater material and technical costs and skills of developers. Along with the requirements for functionality, there are additional requirements for performance, reliability and efficiency of solutions. The choice of the implementation architecture plays a significant role in achieving the requirements for the software systems.

They distinguish an approach whereby a software application is built as a set of small services, each of them works in its own process and communicates with the rest of the services using network mechanisms. Such services are designed to meet specific user needs and deploy independently using a fully automated environment. Microservice-style applications can be written in different languages and use different storage technologies. The increasing complexity of systems and the load on their individual parts require new, more flexible approaches to choosing software development technology, designing its infrastructure, and deploying it.

Microservice architecture is designed for continuous delivery processes. Unlike service-oriented architecture, microservice architecture is aimed to create one or more applications. This takes into account an architectural template and a modular approach that is based on the use of distributed components with defined standard interfaces for application interaction through certain protocols.

Formulation of the task. The problem is to choose a strategy for system architecture design based on monolithic and microservice approaches.

The purpose of this work is to conduct requirements analysis, analysis of existing technologies and software development tools, use of existing systems architecture design approaches.

For further consideration, the task of developing algorithms for creating, deploying and scaling web applications, and analyzing the performance of the developed test system based on measuring the speed of its response.

Research. Designing information systems architecture is one of the most important steps in creating a project. When designing a system, appropriate design decisions are made that affect the behavior of the system and its complexity.

The main goal when choosing a software architecture is to combat the potential complexity inherent in modern software systems, corporate applications. A properly chosen architectural model should be flexible in places that are likely to change or expand most often, but are permanent in others. Also, a properly built system is easy to maintain, test and support. Here are some key points that describe a properly built architecture [1]: system efficiency and reliability, system flexibility, system extensibility, testability, reuse. The main indicators of malformed architecture are rigidity, fragility, high connectivity.

In the development of software architecture, the basic concepts of decomposition and software design are used: the vertical scaling increases the computing power to meet the growing requirements, while the horizontal scaling adds new nodes to the system.

Implementation of all resource services as a single software system involves a monolithic architecture. All services can be implemented using one set of software development technologies and use common code libraries. In

this approach, all services work with a single database server. This allows each service to access the database directly. All request processing logic is executed in a single process.

Most applications deployed to use cloud technologies and any changes require re-assembly and deployment of the entire monolith. Over time, it becomes more difficult to maintain a functional modular structure. Changes in logic of one module tend to affect the code of other modules.

The Service-Oriented Architecture (SOA) architectural approach offers a block system with interacting components instead a monolithic system. In a system built on this principle, the various functional modules of the application are linked through clearly defined interfaces. This approach allows you to distribute the functionality of applications to many services hosted on the network. Web services is a technology for implementing the concept of SOA design. The basic principles of SOA include: weak linking of services, modularity, consistency, repetitiveness, openness, abstraction, one-to-one communication, synchronicity, client process initiation, granularity of services, lack of state [2].

With an Event-Driven Architecture (EDA) approach, the system becomes highly flexible and sensitive to changes in the information environment. The EDA's core components are the event generator (sensor), event handler, and event manager (responder). The basic principles of EDA include: disunity, repetitiveness, real-time messaging, freedom of action, one-to-many communication, synchronicity, recipient process initiation, lack of state of event processing components [2].

Both SOA and EDA architectures break down the process into small, loosely linked procedures that can be reused. These architectural approaches replace and break systems with monolithic architecture into more flexible and reusable components.

When comparing the microservice approach with other existing technologies, a number of determinative advantages are highlighted: increasing the productivity and efficiency of systems by reducing the threshold for parallelism and distribution of individual services, reliability and failure time, sharing functionality and scaling levels as required.

Microservices are a distributed network. Such systems have their own communication tools, including SOAP and REST. The XML standardized protocol (SOAP) for communication does not properly represent the concept of microservice architecture because of the large amount of data in the message, which slows down communication speed between microservices. According to existing requirements, a typical REST service must have a set of identified resources that can be accessed, created, modified and deleted via HTTP (GET type request, POST, PATCH, DELETE). The above communication between microservices has proven itself when working with independent resources.

The situation for the client might be complicated in case of necessity to receive multiple resources from the server and describe those connections in the appropriate format. In addition, microservices interact with each other and are relatively independent entities, so each of them should provide a flexible interface for its clients.

In identifying an effective communication format, the downside of this approach is the lack of standards for messaging format between microservices. The interaction takes place over HTTP using JSON and XML formats, forcing software developers to choose their own message structure.

An example of building a software system based on microservice architecture is considered. The test system allows you to register, get a list of objects, search for the selected criteria and order the selected object. The test system can be conditionally divided into a set of support services (single sign-on point, circuit breaker) and independent standalone microservices. The test system is built on the pattern "Aggregator". Structural elements have been developed to implement the microservice approach in system architecture design.

All interaction between services is synchronous. Each service uses a separate container with a customised environment. To ensure the efficient operation of the system, additional microservices have been introduced that implement microservice architecture templates [3-4].

Load testing is defined as the process of generating demand for system services to measure its response. This approach helped to determine the maximum power of the developed software, to highlight bottlenecks and

problematic elements of the program and infrastructure. The Apache JMeter application (Java-based open source software) was used to perform the load testing and performance measurement task.

Testing of the load in the form of a test on "standard" conditions, ie with normal loading on the system. By measuring the reaction of the system under normal conditions, we obtained a baseline, which was later used in tests to compare with other levels of load. The created sample with N (N = 50) virtual users is used for testing of monolithic and microservice applications, which are formed to solve the same practical problem.

When compared to a monolithic architecture microservice application received about 20% performance. This system is subject to horizontal scaling by enlarging the nodes in the system and meets the required fault tolerance.

Conclusions. The basic concepts of building applications based on microservice architecture are investigated. The main features, advantages and disadvantages of this approach to the construction of software systems in comparison with the classical monolithic solution are considered. The basic principles of designing and deployment of these systems, basic templates for integration of microservices, communication technologies are considered. The test example demonstrates the workability of the concepts considered.

The key components for building microservice systems that form the infrastructure layer and provide the necessary flexibility to the entire system are presented. These components include: single sign service, opening service, balancer, circuit breaker.

One of the main advantages of microservice architecture is its heterogeneity, so have been explored basic libraries to create independent service applications in different programming languages and programming platforms. An important role in the creation of software and its further improvement is played by the support of the necessary environment provided by virtualization systems and continuous integration.

The basic approaches to setting up the processes of continuous integration are analyzed. The test example demonstrates the performance of the above concepts.

REFERENCES

1. Создание микросервисов / Ньюмен С.– СПб.: Питер, 2017– 304 с.
2. Порівняння типів архітектури систем сервісів / О.О.Петренко // Системні дослідження та інформаційні технології. - 2015.-№ 4. - С.48-62.
3. Microservice Architecture: Aligning Principles, Practices, and Culture / I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen – O'Reilly Media, 2016 – 146p.
4. From Design to Deployment / Chris Richardson, Floyd Smith, 2016. – 74 p.

Received 18.12.2019.
Accepted 20.12.2019.

Обрання стратегії проектування архітектури програмного забезпечення

У роботі представлено результати аналізу існуючих методологій проектування архітектури веб-додатків та відповідних інструментальних засобів і ресурсів розробки програмного забезпечення. Виявлено недоліки існуючих підходів щодо розробки додатків та проведено аналіз, який може бути використаний для проектування архітектури веб-систем. Практичною цінністю роботи є розробка алгоритмів створення, розгортання і масштабування додатків за допомогою різних програмних засобів. Розроблено тестову систему на основі декількох класичних підходів.

Selecting the strategy for designing the software architecture

The paper presents the results of the analysis of existing web application architecture design methodologies and related software development tools and resources. The shortcomings of the existing approaches to application development were identified and conducted the analysis can be used to design the web systems architecture. The practical value of the work is the development of algorithms for creating, deploying and scaling applications using various software tools. A test system based on several classic approaches has been developed.

Божуха Лілія Миколаївна - к. ф.-м. н., доцент, Дніпровський національний університет імені Олеся Гончара.
Білобородько Оксана Іванівна - к. т. н., Дніпровський національний університет імені Олеся Гончара.

Божуха Лилия Николаевна - к. ф.-м. н., доцент, Днепровский национальный университет имени Олеся Гончара.
Белобородько Оксана Ивановна - к. т. н., Днепровский национальный университет имени Олеся Гончара.

Bozhukha Liliia - k. f.-m. n., associate professor, Dnipro National University named after Oles Honchar.
Beloborodko Oksana - Ph.d., Dnipro National University named after Oles Honchar.