RESEARCH ARTICLE OPEN ACCESS

ARCHITECTURAL PATTERNS IN DISTRIBUTED SYSTEMS: MICROSERVICES, EVENT-ORIENTED SYSTEMS, AND CQRS

Glumov Konstantin
Senior Software Engineer, Alfa-Russia, Perm
kglumov@alfabank.ru

Abstract. The article discusses architectural patterns in distributed systems: microservices, event-driven systems, and CQRS. The research problem lies in the fact that during software creation, developers are faced with overloaded system code because the written code must be readable, which is why developers add the necessary details and explanations to the program code listing. The review of literature sources concerning the information systems architecture was performed, a multidisciplinary review of microservice architecture patterns was carried out, the CQRS pattern application features in event-driven architecture of high-volume distribution systems were considered, and the design features of microservice-event-driven architectures for high-volume distribution information processing systems were reviewed, which made it possible to conclude that insufficient attention had been paid to the architectural patterns in distribution systems in the literature, which points out the importance of research in this direction. Such architectural approaches to software development as microservices, event-driven systems, and CQRS were considered. Architectural patterns for software development approaches were reviewed. The study results show that architectural approaches and patterns are tools, the choice of which depends only on the developers, who must be guided by such selection criteria as efficiency, benefits, costs, and rationality. The conclusions drawn from the study may be useful to software developers who develop distributed systems to speed up information systems development, improve software quality, and simplify its support.

Keywords: patterns, architecture, CQRS, distributed systems, microservices, event-driven systems.

1. Introduction

During software creation, developers face the problem of voluminous application code. This is because the written code has to be readable, due to which developers add necessary details and explanations in the program code listing. Ready templates, namely patterns, in applications eliminate code overflow.

Application architecture provides application usability and extensibility, due to which choosing the optimal architecture is a non-trivial task. Significant aspects of architecture selection include detailed analysis of software feature requirements, development team experience, business requirements, and project timeframe.

2. Literature Review

The work of M.V. Rybalchenko is devoted to the study of information systems architecture [1], a study by Kh.A. Valdia [2] provides a multidisciplinary review of microservice architecture patterns. The peculiarities of CQRS pattern application in the event-driven architecture of highly loaded distribution systems are considered in the works of N.O. Khitrov and M.A. Gorbachev [3]. The peculiarities of designing microservice-event-oriented architectures for highly loaded distribution systems of information processing are presented by M.N. Karpovich in his study. [4].

The consideration of architectural patterns in distribution systems has not received enough attention in the literature, suggesting the importance of research in this direction. The study aims to review architectural patterns in such distributed systems as microservices, event-driven systems, and CQRS.

3. Methodology

Today, some of the most popular architectural approaches to software development are microservices, event-driven systems, and CQRS (Figure 1).

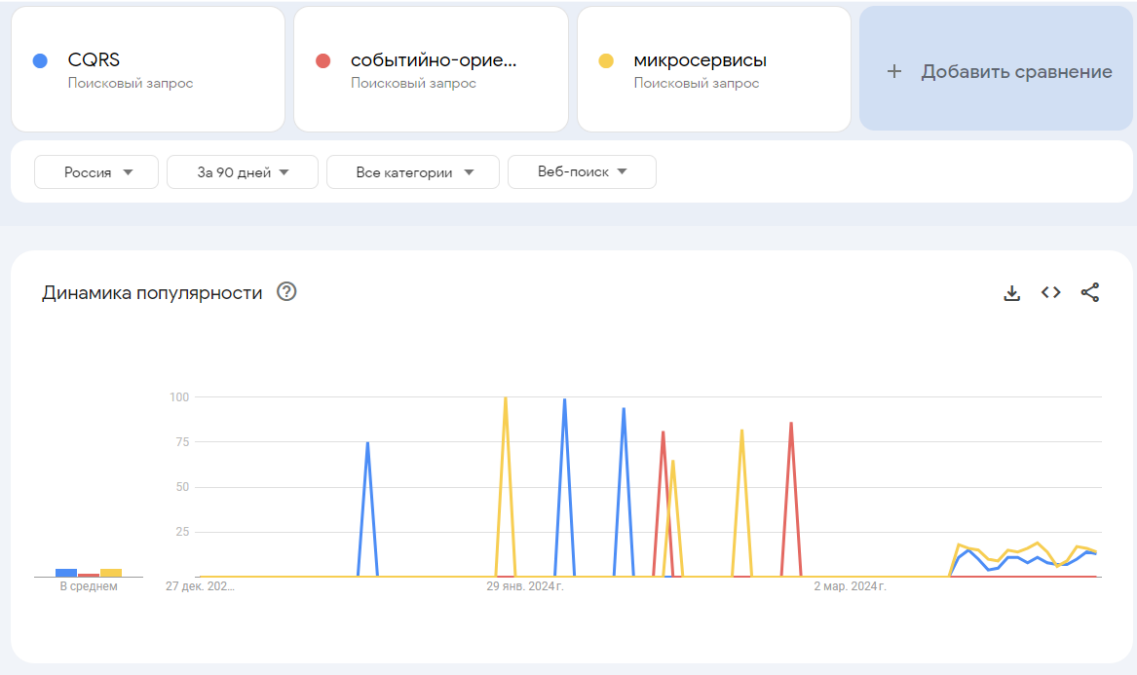

Figure 1 - Popular architectural approaches to software development

Microservices is an architectural approach to information systems development, which means that the application consists of several independent services that are responsible for individual functions. These functions interact with each other due to certain APIs.

Advantages of microservice architecture:

- convenient application scalability;

- improved application security;
- resistance to high parallelism levels;
- relatively low level of labor costs.

Besides, if some microservice architecture module fails or is attacked, the program will not become inaccessible; it will continue to function, and only the functionality of the damaged module will become inaccessible.

Disadvantages of microservice architecture:

- high maintenance requirements;
- significant investment costs;
- improved application security;
- high operation requirements.

Event-driven architecture is a software design pattern that involves events to trigger and pass changes between application components.

Advantages of event-driven architecture:

- ease of scalability since the software will consist of a few asynchronous modules;
- high service performance.

Disadvantages of event-driven architecture:

- complexity of debugging, due to the asynchronous module nature;
- possibility of failures when launching several chains of actions;
- logging complexity;

CQRS represents an architectural approach to information systems design that involves separating reading and writing operations to realize application scalability and security enhancements. This approach uses commands to write data to persistent storage and queries to discover and retrieve data. These commands and queries are handled by a control center that receives requests from users and subsequently executes them, stores them, and notifies the procedure.

Advantages of CQRS architecture:

- separation of responsibility;
- small number of unpredictable changes to distributed data;
- application complexity reduction through delegation;
- reduction in the number of data-modifying entities;
- realizes a clear separation of business logic and validation.

Disadvantages of CQRS architecture:

- latency potential when sending a large number of queries;
- interaction between teams and queries continuously;
- no means of communication between service processes.

4. Research and Results

Microservice architectural patterns can create fault-tolerant, efficient, and easily scalable systems. Today, there are a large number of patterns available for distributed systems, but consider the most popular ones.

The Universal IT Solution for API Gateway Pattern microservices (Figure 2) is a pattern that can be used as a single point of entry for client requests, thus providing seamless interaction between services and clients.

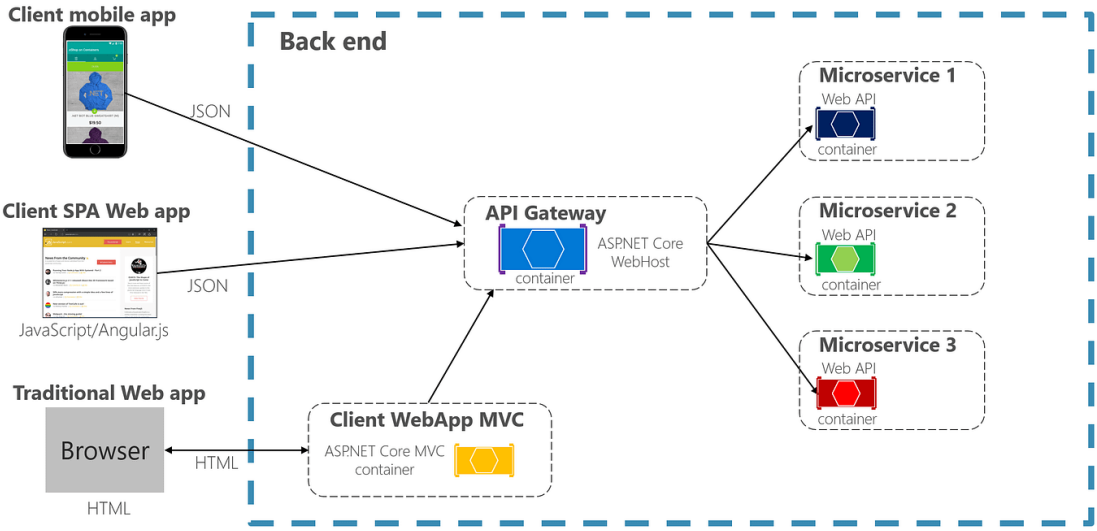

Figure 2 - API Gateway Pattern

The API Gateway Pattern aggregates responses from multiple microservices, reducing the number of requests between clients and services, which will result in improved performance. This pattern also allows common functions to be organized in one place, resulting in consistency and reduced redundancy.

Another popular pattern is the Strangler Pattern (Figure 3), which replaces a monolithic system with microservices while ensuring a smooth and secure transition. Since the transition from monolithic architecture to microservice architecture is a complex and risky process, the use of Strangler Pattern will enable implementing a phased replacement, minimizing downtime and preserving business continuity.

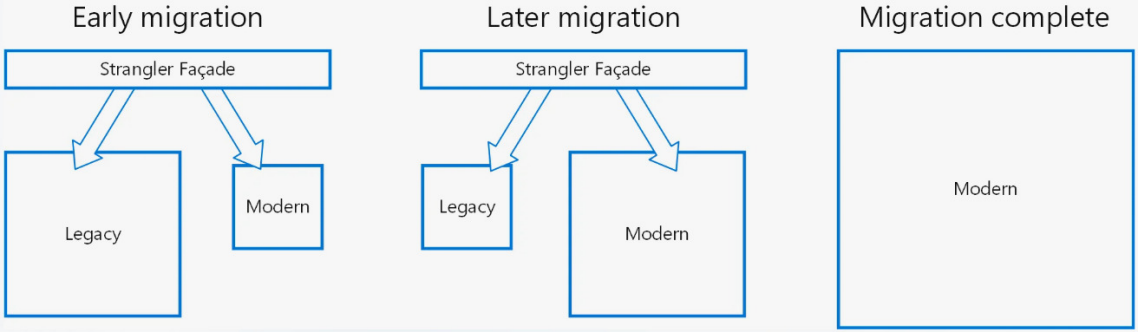

Figure 3 - Strangler Pattern

Service Discovery Pattern (Figure 4) is a pattern that enables modules to search for each other, ensuring seamless communication and reducing manual configuration processes.

This pattern is important because as the application scales up, the module management becomes more complex, and using the pattern will allow the modules to react and search for each other automatically, increasing the service's flexibility and scalability.

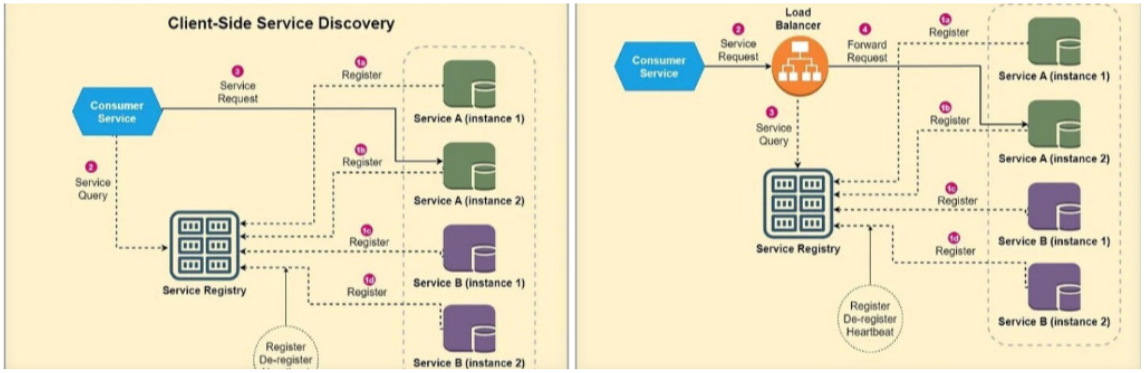

Figure 4 - Service Discovery Pattern

The Circuit Breaker Pattern (Figure 5) is a pattern that protects against cascading failures. It constantly monitors service failures and prevents requests from reaching a service that is down for the entire system's recovery and protection.

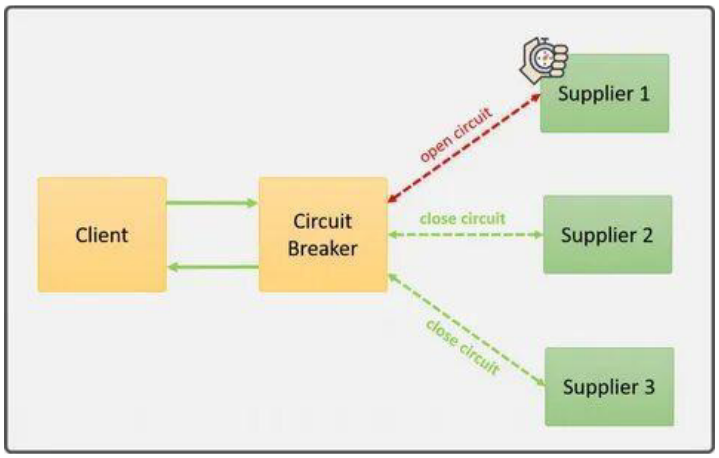

Figure 5 - Circuit Breaker Pattern diagram

This pattern is important because if a service in the application is not functioning properly, the domino principle can be triggered, i.e., services that do not depend on each other can be disrupted. The Circuit Breaker Pattern helps isolate the incorrectly functioning service, which will prevent further system destruction.

The Event-Driven Architecture Pattern is a pattern designed to trigger events in services for a real-time response. Figure 6 shows a diagram of the Event-Driven Architecture Pattern.

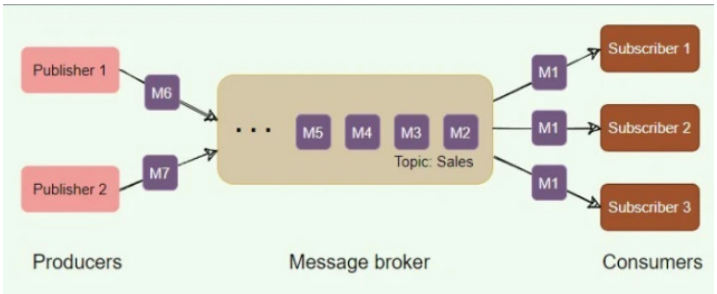

Figure 5 - Event-Driven Architecture Pattern diagram

The EDA uses events as triggers that minimize direct dependencies between services, which in turn leads to increased flexibility and easier service scaling.

The CQRS Pattern is a pattern that aims to improve performance by separating tasks (Figure 6). This pattern separates reading and writing services, allowing the administrator to configure each aspect in a way that maximizes the impact.

Traditional architectures combine reading and writing operations, which can result in degraded application performance as well as increased complexity. CQRS allows each operation to be optimized separately, which will have the opposite effect, i.e., will increase performance and simplify maintenance.

This pattern's implementation will divide the service into two parts. The first part will deal with the processing of writing commands, and the second part will deal with the processing of reading requests. Thus, this division helps use different scaling, caching, and database strategies for each operation type.

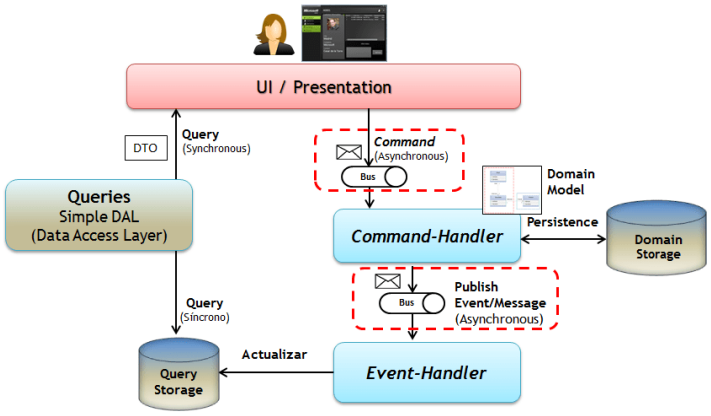

Figure 7 - CQRS Pattern

Thus, it can be concluded that the need for fault-tolerant, efficient, and easily scalable systems is of paramount importance because, in the information technology era, preferences for information systems development environments are rapidly changing.

Conclusion

Each of the considered architectural approaches and patterns is a means of solving certain problems, and therefore, software developers do not need to treat them as a goal when writing code since there are cases when an IT solution without a complex architecture is simpler and more efficient.

Architectural approaches and patterns are tools, the choice of which depends only on the developers, who must be guided by such selection criteria as efficiency, benefits, costs, and rationality.

References

1. Rybalchenko, M.V. Architecture of information systems: textbook for open source education / M.V. Rybalchenko. – M.: Yurayt, 2021. – 91 p.

2. Valdivia H.A. Microservice architecture patterns: a multidisciplinary literature review / Kh.A. Valdivia, A. Laura-Gonzalez, C. Limón, C. Cortes-Verdin., J.O. Ocharan-Hernandez // Proceedings of ISP RAS. – Mexico: University of Veracruz, 2021. – P.81-96

3. Khitrov, N.O. Features of the use of the CQRS pattern in the event-driven architecture of high-load distributed systems / N.O. Khitrov, M.A. Gorbachev // Scientific and educational magazine for students and teachers “StudNet” - M.: RTU, 2021. - P.1210-1216

4. Karpovich, M.N. Features of designing microservice-event architectures for distributed information processing systems // Proceedings of BSTU. Series 3: Physical and mathematical sciences and computer science. – Minsk: BSTU, 2023. – P. 89-95

