УДК 004.4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕАЛІЗАЦІЇ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ ДЛЯ РОЗПОДІЛУ ФУНКЦІЙ ПРОЕКТУ СИСТЕМИ ОБЛІКУ ІВЕНТ-ПОСЛУГ

Ходирєв Є. О.
Науковий керівник – к.т.н., с.н.с. Коваленко А. І.
Харківський національний університет радіоелектроніки, каф. СТ
м. Харків, Україна
email: yehor.khodyriev@nure.ua

This work is devoted to the analysis of the methods of implementing microservice architecture, which allows to ensure a high level of scalability, reliability and flexibility in the management of the information system. During the study, several approaches to the development of microservices were considered the use of containerization and orchestration of containers. Key microservices were defined for the event agency's order accounting system, such as customer accounting service, event planning service, animator accounting service, and others. Each microservice is developed and deployed independently, which provides the ability to quickly make changes and optimize individual components of the system without affecting the operation of other services.

На сучасному етапі розвитку інформаційних систем особлива увага приділяється підвищенню ефективності та гнучкості бізнес-процесів, зокрема в сфері надання івент-послуг. Одним із перспективних напрямків оптимізації роботи івент-агентств є впровадження мікросервісної архітектури, це дозволить розподіляти функції проекту системи обліку івент-послуг між незалежними сервісами.

Ця доповідь присвячена аналізу методів реалізації мікросервісної архітектури, яка дозволяє забезпечити високий рівень масштабованості, надійності ті зручності в управлінні інформаційною системою. В ході дослідження було розглянуто кілька підходів до розробки мікросервісів, зокрема, використання контейнерізації та оркестрації контейнерів.

Для системи обліку замовлень івент-агентства було визначено ключові мікросервіси, такі як сервіс обліку клієнтів, сервіс планування заходів, сервіс обліку аніматорів та інші. Кожний мікросервіс розробляється та розгортається незалежно, що забезпечує можливість швидкого внесення змін та оптимізації окремих компонентів системи без впливу на роботу інших сервісів.

Одним із ключових аспектів реалізації мікросервісної архітектури в інформаційній системі обліку замовлень івент-агенстства є забезпечення ефективної взаємодії між окремими сервісами. В ході дослідження були розглянуті різні методи комунікації між мікросервісами, зокрема синхронні та асинхронні підходи. Синхронна комунікація, зазвичай

реалізована через HTTP/REST запити, забезпечує прямий та негайний обмін даними, це важливі критерії для операцій, які вимагають миттєвої відповіді. Асинхронна комунікація, здійснюється через механізми черг повідомлень або подій, це дозволяє сервісам обмінюватися даними без необхідності відповідати в реальному часі, що підвищує гнучкість та масштабованість системи.

Для з'єднання точок доступу різних мікросервісів та спрощення взаємодії з клієнтською стороною використовуються API шлюзи. Ці шлюзи діють як єдина вхідна точка, що приймає запити від клієнтів та перенаправляє їх до відповідних мікросервісів, а також агрегує результати від різних сервісів для відправки назад клієнту. Використання API шлюзів не тільки спрощує архітектуру системи з точки зору клієнта, але й забезпечує додатковий рівень абстракції, який може використовуватися для реалізації безпеки, моніторингу, обмеження швидкості запитів та інших загальносистемних функцій. Отже, вивчення та впровадження ефективних методів взаємодії між мікросервісами є важливим елементом для створення гнучкої, масштабованої та надійної системи обліку івент-послуг.

Розробка мікросервісів проводилася з використанням фреймворку Spring Boot[1], необхідного для швидкої розробки високопродуктивних мікросервісів з простим впровадженням залежностей та управлінням конфігурацією. Для контейнеризації сервісів використовувалась платформа Docker[2], а оркестрація контейнерів забезпечувалась програмою Kubernetes, що дозволило автоматизувати розгортання, масштабування та управління мікросервісами.

У системі обліку івент-послуг, реалізованій на основі мікросервісної архітектури, всі сервіси будуть використовувати NoSQL[3] базу даних. Це забезпечить гнучкість у структурі даних та підвищить швидкість обробки запитів, що є важливим для ефективної роботи кожного сервісу в системі.

Вибір NoSQL бази даних дозволяє кожному мікросервісу оптимізувати свою роботу з даними, адаптуючись під специфічні потреби без необхідності узгодження схеми даних з іншими сервісами. Це полегшує процес розробки та впровадження нових функцій у систему.

Список використаних джерел:

1. Spring Boot : вебсайт. URL: https://spring.io/projects/spring-boot (дата звернення: 27.02.2024).

2. Docker and Kubernetes : How They Work Together // Alyssa Shames: блог. URL: https://www.docker.com/blog/docker-and-kubernetes (дата звернення: 27.02.2024).

3. NoSQL : вебсайт. URL: https://aws.amazon.com/nosql (дата звернення: 27.02.2024).