УДК 004

Розбицький Р.Е., магістрант
Бродський Ю. Б, к.т.н., доцент
Державний університет «Житомирська політехніка»

АНАЛІЗ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ ДЛЯ РОЗРОБКИ МАСШТАБОВАНИХ ВЕБ-СИСТЕМ

Мікросервісна архітектура (МСА) є підходом до побудови програмних систем, що грунтується на розподілі застосунку на невеликі, прямо не пов'язані сервіси, кожен із яких реалізує окрему бізнес-функцію. На відміну від монолітних систем, де всі модулі працюють у межах одного середовища, мікросервіси функціонують автономно та взаємодіють через легковагові мережеві протоколи. Цей підхід набув популярності стрімкому розширенню великих веб-застосунків, які зіштовхнулися з потребою відокремлення функціональності для підвищення ефективності розробки, стабільності системи та покращення масштабованості[1].

Однією з ключових особливостей МСА є незалежність сервісів. Кожен із них має власний життєвий цикл, кодову базу, середовище виконная та розгортання. Тобто кожен сервіс вважається окремим програмним додатком. Декомпозиція на дрібні компоненти дає змогу командам розробників працювати паралельно, не блокуючи одна одну, що пришвидшує створення функціоналу та зменшує ризик конфліктів під час інтеграції компонентів системи між собою[2]. Мікросервіси також часто характеризуються принципами polyglot persistence та polyglot programming: кожен сервіс може використовувати різні мови програмування та бази даних, відповідно до своїх потреб. Це підвищує оптимальність вибору технологій, але водночас ускладнює загальну інфраструктуру[3].

Важлвою технічною особливістю мікросервісної архітектури є розподілений характер системи. Мікросервіси взаємодіють через протоколи HTTP, REST, gRPC або черги повідомлень, що забезпечує гнучкість, але вводить додаткові мережеві ризики: затримки, втрату пакетів, непередбачувану поведінку. Унаслідок цього проєктування МСА невіддільне від практик DevOps, контейнеризації, оркестрації та автоматизованого моніторингу. Без інструментів на кшталт Docker, Kubernetes, систем логування й трасування, мікросервісна система швидко стає некерованою[4].

Серед основних переваг МСА виділяють високу масштабованість: кожен сервіс можна збільшувати або зменшувати незалежно від інших, що дає змогу ефективно розподіляти як людські, так і технічні ресурси. Ізоляція сервісів підвищує стійкість до помилок

— збій одного компоненту великої системи не призводить до її цілковитого падіння. Завдяки високій модульності спрощується підтримка, перероблення та впровадження нового функціоналу, а також створюються умови для впровадження гнучких методологій та постійного розгортання (CI/CD). Багато досліджень свідчать, що перехід до МСА покращує швидкість релізів та зменшує ризик системних збоїв, що особливо важливо для великих високонавантажених продуктів[5].

Разом із тим мікросервісна архітектура має певні недоліки. Найсуттєвішим є значне ускладнення інфраструктури: з'являється потреба в балансуванні навантаження, централізованому логуванні, тонкому налаштуванню API, підтримці стабільності даних та вирішенні проблем розподілених транзакцій в базах даних. Вартість розробки та підтримки системи може зрости, оскільки навіть прості операції потребують налагодженої взаємодії між сервісами. Наостанок, через більш високу абстрактність та загальну технічну складність, мікросервісна архітектура потребує зрілої команди розробників з високими технічними компетенціями.

Таким чином, мікросервісна архітектура є потужним підходом для великих, складних та швидко зростаючих програмних продуктів, для яких пріорітетом є гнучкість, масштабованість, простота підтримки та висока стійкість до помилок. В результаті аналізу визначено що мікросервісна архтектура дійсно є оптимальним архітектурним підходом до розробки великих масштабованих веб-систем, але підходить не для всіх систем меншого масштабу. Ми пропонуємо використовувати мікросервісну архітектуру за наявності досвідченої команди розробки та сучасної технічної бази.

Список використаних джерел:

1. Newman S. *Building Microservices.* 2nd ed. Sebastopol: O'Reilly Media, 2021. 352 c..

2. NGINX Inc**.** *Microservices Reference Architecture: NGINX Whitepaper* [Електронний ресурс]. 2021. Режим доступу: https://www.nginx.com

3. IBM Corporation. *Microservices Guide 2023: Principles, Patterns, and Deployment Models* [Електронний ресурс]. 2023. Режим доступу: https://www.ibm.com/cloud/architecture

4. Microsoft Azure Architecture Center. *Microservices Architecture Style – Updated Best Practices 2022* [Електронний ресурс]. 2022. Режим доступу: https://learn.microsoft.com/azure/architecture

5. Dragoni N., Giallorenzo S., Lafuente A. L., et al. *Microservices: Migration and Architectural Perspectives – 2020 Update.* ACM Digital Library, 2020. DOI: 10.1145/3380768.3380775.