УДК 004.4

А. Б. Форостяний
Н. П. Бабюк

ЕВОЛЮЦІЯ АРХІТЕКТУРНИХ СТИЛІВ: ВІД МОНОЛІТУ ДО МІКРОСЕРВІСІВ

Вінницький національний технічний університет, Україна

Анотація

У роботі розглянуто перехід від монолітної архітектури до мікросервісної та гібридної. Проаналізовано їхні особливості, переваги, недоліки, а також сучасні технології взаємодії між сервісами. Наведено приклади використання гібридних рішень у великих компаніях та акцентовано увагу на важливості вибору архітектури відповідно до потреб проєкту.

Ключові слова: монолітна архітектура, мікросервісна архітектура, гібридна архітектура, масштабованість, подієво-орієнтована архітектура, розробка програмного забезпечення.

Abstract

This text examines the shift from monolithic to microservices and hybrid architectures. It analyzes their features, pros and cons, and the modern technologies used for service communication. Examples from large companies are provided, emphasizing the importance of selecting architecture based on project needs.

Keywords: monolithic architecture, microservices architecture, hybrid architecture, scalability, event-driven architecture, software development.

Вступ

Розвиток програмного забезпечення супроводжується зміною архітектурних підходів. Одним із найважливіших етапів цієї еволюції є перехід від монолітної архітектури до мікросервісної. У цій тезі розглянуто особливості кожного підходу, їхні переваги та недоліки, а також виклики, що виникають при переході на мікросервіси.

Монолітна архітектура

Монолітна архітектура [1] (Monolithic Architecture) – це традиційний підхід до розробки програмного забезпечення, при якому весь додаток розробляється як одна єдина технологічна система. Всі компоненти додатку взаємодіють один з одним і розгортання відбувається на одному сервері або групі серверів. Ця архітектура передбачає єдину кодову базу, де всі компоненти тісно пов’язані між собою. Саме тому вона зазвичай простіша в розробці та тестуванні, оскільки всі компоненти додатку взаємодіють безпосередньо один з одним, що дозволяє швидко та легко вирішувати проблеми. Крім того, монолітні додатки можуть бути менш складними в управлінні, оскільки все програмне забезпечення працює на одній технологічній платформі. Однак монолітна архітектура має недолік – зі збільшенням обсягу застосунку можуть виникати обмеження в його гнучкості та масштабованості. Також при внесенні правок може знадобитися зміна всього кодового базису, а це тривалий і трудомісткий процес, а при несправній роботі одного компонента може вийти з ладу навіть ціла система.

Перехідний етап

Щоб зробити систему більш гнучкою, компанії почали виділяти окремі компоненти в автономні сервіси, які взаємодіють через API. Це можна розглядати як перший крок до мікросервісної архітектури. REST API [2] став одним із найпоширеніших способів організації взаємодії між сервісами. Він базується на протоколі HTTP та використовує стандартні методи, такі як GET (отримання даних), POST (створення ресурсу), PUT (оновлення ресурсу) та DELETE (видалення ресурсу). RESTful-сервіси легко інтегруються з веб-додатками, підтримуються більшістю мов програмування та можуть працювати у розподілених системах. Однак REST API має певні обмеження, наприклад, він не оптимізований для високопродуктивних сервісів, які потребують швидкої взаємодії

та передачі великих обсягів даних. Для підвищення продуктивності та ефективності обміну даними часто використовують gRPC [3]. На відміну від REST, де дані передаються у форматі JSON або XML, gRPC дозволяє стиснути інформацію та значно зменшити затримки під час передачі. Це особливо корисно для високонавантажених сервісів, які потребують швидкого обміну даними між компонентами. Крім того, gRPC підтримує автоматичне генерування клієнтів і серверів для різних мов програмування, що спрощує інтеграцію в багатомовних системах.

Мікросервіси

Мікросервісна архітектура [4], на відміну від монолітної, поділяє систему на багато дрібних автономних частин (мікросервісів), де кожен виконує окрему функцію. Це фактично міні-програми. Ці частини взаємодіють між собою через спеціальні протоколи (наприклад, через API). Компанії переходять від монолітів до мікросервісів через необхідність гнучкості, швидкого оновлення програмного забезпечення та ефективного розподілу навантаження. Прикладом такого мікросервісу може бути інтернет-магазин, який має окремий мікросервіс для управління товарами, інший для обробки замовлень і ще один для роботи з користувачами. Ці компоненти працюють незалежно один від одного та водночас разом забезпечують функціонування магазину. Варто пам’ятати, що управління мікросервісами потребує розвинених механізмів безпеки, моніторингу та тестування.

Гібридні архітектури стали популярним рішенням для компаній, які хочуть зберегти переваги монолітного підходу, водночас використовуючи гнучкість мікросервісів. У таких системах критично важливі компоненти, що потребують високої продуктивності, стабільності та узгоджених транзакцій, залишаються в межах монолітної архітектури, тоді як допоміжні сервіси, наприклад, обробка платежів, аналітика або рекомендаційні системи, реалізуються як мікросервіси. Це дозволяє поступово рефакторити код, спрощуючи міграцію від моноліту до мікросервісів. Щоб забезпечити ефективну взаємодію між компонентами, широко використовуються подієво-орієнтовані підходи, такі як Kafka [5], RabbitMQ [6] або AWS EventBridge [7], які дозволяють асинхронний обмін повідомленнями між сервісами. Наприклад, Netflix використовує гібридний підхід, де більшість критичних функцій залишаються централізованими, але окремі сервіси (такі як управління контентом і персоналізація) працюють на основі мікросервісів та подій. Amazon поєднує мікросервіси з монолітними сервісами у своїх логістичних і фінансових системах, використовуючи подієво-орієнтовану архітектуру для обробки транзакцій у реальному часі. Spotify також реалізував гібридну модель, де монолітне ядро взаємодіє з незалежними сервісами, які відповідають за рекомендації, рекламу та інтеграцію з іншими платформами. Таким чином, гібридні архітектури допомагають компаніям досягти балансу між продуктивністю, надійністю та масштабованістю, адаптуючись до постійних змін бізнес-вимог.

Висновки

Отже, еволюція архітектурних підходів від монолітних систем до мікросервісних і гібридних рішень є природним процесом, спричиненим зростаючими вимогами до масштабованості, гнучкості та ефективності програмного забезпечення. Монолітна архітектура залишається актуальною для невеликих і середніх проєктів, тоді як мікросервіси забезпечують кращу адаптивність у великих, динамічних системах. Гібридні моделі стають компромісним варіантом, поєднуючи переваги обох підходів і дозволяючи компаніям поступово переходити до децентралізованих рішень. Вибір оптимальної архітектури залежить від конкретних бізнес-вимог, технологічних можливостей і ресурсів, необхідних для підтримки та розвитку системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Монолітна архітектура ПЗ – URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/.
2. REST API як спосіб спілкування компонент веб-додатків – URL: https://foxminded.ua/shcho-take-rest-api/.
3. About gRPC – URL: https://grpc.io/about/.
4. Основні типи архітектури програмного забезпечення – URL: https://www.artofba.com/uk/post/main-types-of-software-architecture.
5. Apache Kafka – URL: https://openhub.net/p/apache-kafka/analyses/latest/languages_summary.
6. RabbitMQ – URL: https://www.rabbitmq.com/.
7. Amazon EventBridge – URL: https://aws.amazon.com/eventbridge/.

Форостяний Артур Богданович – студент групи 5ПІ-23б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: bforostyaniy@gmail.com

Науковий керівник: Бабюк Наталя Петрівна – доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, e-mail: babiuk@vntu.edu.ua

***Forostianyi Artur B.* –** Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: bforostyaniy@gmail.com

Supervisor: Babiuk Natalia P. – Associate Professor of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: babiuk@vntu.edu.ua