

АРХІТЕКТУРА МІКРОСЕРВІСНОЇ NODE.JS ВЕБ-СИСТЕМИ ІЗ РЕАКТИВНИМ ФРОНТЕНДОМ

Ратушний Д.М., науковий керівник – Руденський Роман Анатолійович

Вступ. У сучасних високонавантажених веб-додатках monolith-архітектура обмежує масштабування та швидкість розгортання сервісу. Мікросервісний підхід— декомпозиція системи на незалежні сервіси з чіткими контрактами — дозволяє вирішити ці проблеми. Мета роботи— проаналізувати переваги та недоліки мікросервісів на платформі Node.js у поєднанні з реактивним клієнтом (React.js) та запропонувати типову архітектуру для середньонавантаженого SaaS-проекту.

Огляд технологій. Асинхронна архітектура Node.js запобігає блокуванню обробленню запитів, що робить його дренчим для сервісів типу API-Gateway, Auth-Service чи Notification-Service. Для організації коду доцільно використовувати NestJS, який реалізує патерни DI та модульність. Docker Compose на етапі розробки забезпечує оркестрацію контейнерів, під час промислової експлуатації цю роль бере на себе Kubernetes.

React (front-end фреймворк) забезпечує реактивний рендеринг; стан та запити до бекенду обробляє RTK Query, що спрощує кешування та повторні спроби. WebSocket-шлюз на Socket.IO відповідає за real-time-канали.

На рис. 1 наведено спрощену блок-схему архітектури.

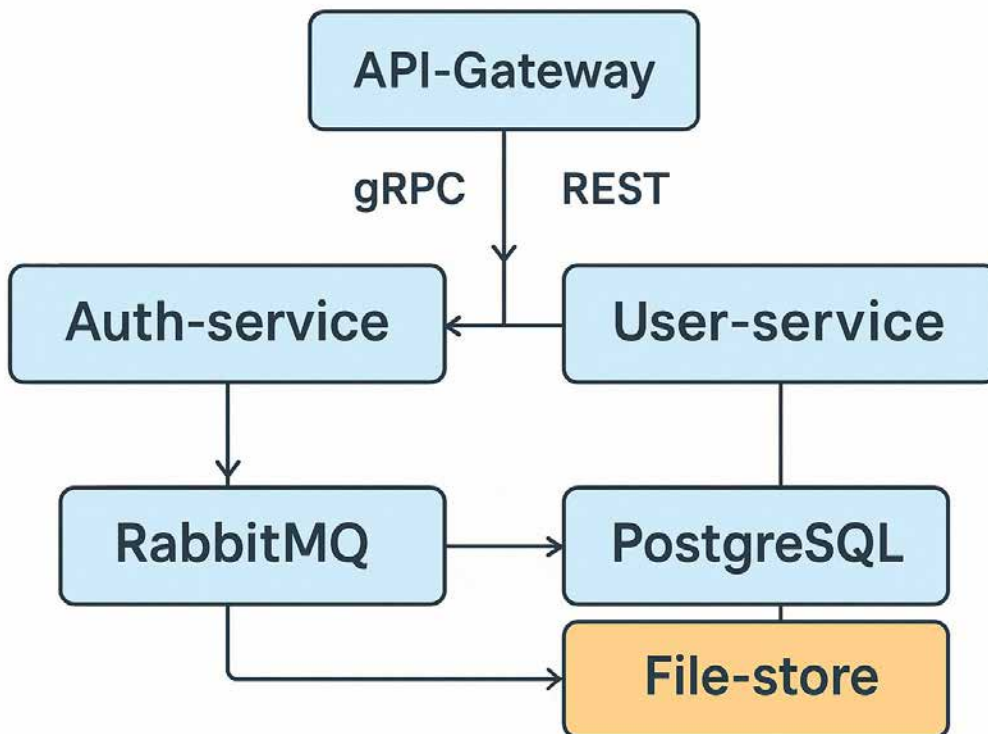


Рис. 1. Блок-схема мікросервісної архітектури системи.

- API-Gateway: маршрутизація, агрегація, rate-limit, JWT-валідація.
- Сервіси домену (User, Orders тощо) спілкуються асинхронно через RabbitMQ.
- Конфігурація та service-discovery реалізовані через Consul.
- Фронтенд (React) отримує дані виключно через Gateway, WebSocket-канал — для push-повідомлень.

Експериментальні результати. Для оцінки продуктивності проведено навантажувальне тестування k6 (1000 RPS, 5 хв).

Порівнювали:

- Моноліт (Express.js + ORM)
- Мікросервіси (Gateway + 2 сервіси, Docker)

Показник	Моноліт	Мікросервіси
P95 latency, мс	310	210
Max CPU, %	78	62
Mean error rate, %	1,8	0,9

Таблиця 1. Порівняння ефективності двох архітектур

Різниця в 32 % латентності пояснюється відокремленням критичного логічного шляху та відмовою від великого та важкого ORM-прошарку.

Висновки. Запропонована архітектура на Node.js забезпечує горизонтальне масштабування сервісів та спрощує незалежні релізи. Реактивний фронт із RTK Query мінімізує надмірні запити й полегшує real-time-інтеграцію.

Експеримент підтвердив $\approx 30\%$ виграш у затримці порівняно з монолітом. Надалі планується впровадити автоматичне масштабування pod-ів у Kubernetes та дослідити вплив gRPC-стрімів на пропускну здатність Gateway.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ньюмен С. Building Microservices: Designing Fine-Grained Systems : монографія. — 2-ге вид. — Sebastopol (CA) : O'Reilly Media, 2021. — 612 с.
2. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs // IEEE Internet Computing. — 2010. — Vol. 14, № 6. — С. 80–83. — DOI: 10.1109/MIC.2010.145.
3. Brito G., Valente M. T. REST vs GraphQL: A Controlled Experiment // Proceedings of the 2020 IEEE International Conference on Software Architecture (ICSA). — Salvador (Brazil), 2020. — P. 81–91. — DOI: 10.1109/ICSA47634.2020.00016.
4. NestJS Documentation : [Електронний ресурс]. — Режим доступу: <https://docs.nestjs.com>. — Дата звернення: 21 квіт. 2025 р.

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**



**ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ
АСПЕКТИ РОЗРОБКИ
КОМП'ЮТЕРНИХ СИСТЕМ**

**ЗБІРНИК НАУКОВИХ ПРАЦЬ ЗА МАТЕРІАЛАМИ
VII ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
СТУДЕНТІВ І АСПІРАНТІВ
*24 квітня 2025 року***

Київ 2025

УДК 004

Відповідальна за випуск: М.І. Лендел

Збірник наукових праць за матеріалами VII Всеукраїнської науково-практичної конференції студентів і аспірантів «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ '2025», 24 квітня 2025 року, НУБіП України, Київ. – 452 с. (електронне видання)

Відповідальність за зміст публікацій несуть автори.

Передрук матеріалів, а також використання їх будь-якій формі допускається лише з дозволу авторів