DOI: 10.34185/1991-7848.itmm.2026.01.041

ДЕФІЦИТ УВАГИ НАУКОВОЇ СПІЛЬНОТИ ДО ПРОБЛЕМАТИКИ API В ЕПОХУ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Кушнір Б. Т.[1] [ORCID], Іванов О. П.[2] [ORCID]

[1]*Український державний університет науки і технологій, аспірант, Україна*

[2]*Український державний університет науки і технологій, к.т.н., доцент, Україна*

Анотація. *Досліджено аномалію сучасного наукового дискурсу щодо мікросервісної архітектури – систематичне недооцінювання важливості ролі інтерфейсів прикладного програмування (API) на тлі домінування публікацій з хмарних обчислень, штучного інтелекту та інших перспективних напрямків. Проведений аналіз публікацій у провідних наукометричних базах даних за період 2020–2025 рр. засвідчив суттєвий дисбаланс: обсяг досліджень мікросервісної архітектури та суміжних технологій значно перевищує кількість робіт, присвячених безпосередньо менеджменту, еволюції та дизайну програмних інтерфейсів. API виступає ключовим елементом мікросервісів, без якого компоненти залишаються ізольованими, а система – нефункціональною. Виявлено прогалини в науковому дискурсі, зокрема недостатню розробку методологій сталого розвитку API, проблеми зворотної сумісності та управління несумісними змінами.*

Ключові слова: *програмне забезпечення, мікросервісна архітектура, еволюція API, зворотна сумісність.*

Метою даної роботи є проведення аналізу наукової літератури, виявлення прогалин у дослідженні проблематики API в контексті мікросервісної архітектури та формулювання науково обґрунтованих пропозицій щодо подолання виявленого дефіциту.

Для досягнення поставленої мети було виконано аналіз публікацій у провідних наукових базах даних, зокрема Google Scholar, IEEE Xplore, ACM Digital Library, ResearchGate та arXiv, за період 2020–2025 рр. Ключові пошукові запити включали «Microservices API evolution», «API breaking changes», «API maintenance costs», «REST API versioning microservices». Додатково було залучено результати третинних досліджень, зокрема роботи Stojanov та інших [1], які проаналізували 44 вторинні дослідження з мікросервісної архітектури.

Результати пошуку демонструють суттєву асиметрію в науковому дискурсі. Кількість публікацій з мікросервісної архітектури в цілому сягає десятків тисяч, тоді як робіт, присвячених безпосередньо менеджменту та дизайну API, – лише кілька сотень. При цьому тематика еволюції API та несумісних змін представлена ще меншою кількістю публікацій. Серед основних тем досліджень мікросервісної архітектури домінують архітектурний дизайн, життєвий цикл, міграція та виклики впровадження. Однак питання API в цих роботах розглядаються переважно побічно, без глибокого аналізу специфічних проблем.

Найбільш релевантними дослідженнями з проблематики API в мікросервісній архітектурі є роботи Lercher [2], присвячені управлінню еволюцією API. Виявлено, що команди розробників неформально комунікують зміни REST API, а також проблеми з несумісними змінами та міграцією споживачів на нові версії. Дослідження Collina, Maraschi та Pirini [3] запропонувало алгоритм для визначення ризику, пов'язаного з кожною зміною будь-якого мікросервісу в системі, при цьому автори зазначають, що здатність незалежно розгортати сервіси водночас ускладнює управління ризиками, оскільки будь-яка модифікація сервісу може спричинити відмову всієї системи. Lahtinen [4] з Гельсінського університету присвятила дослідження спрощеному методу виявлення пошкоджень API в мікросервісних архітектурах. Yaroshynskyi та інші [5] дослідили проблеми та ризики, пов'язані зі змінами API під час еволюції програмного забезпечення.

Аналіз літератури дозволив виділити основні прогалини в науковому дискурсі:

- більшість публікацій фіксує проблеми методологій сталого розвитку API, але не пропонує системних рішень;
- питання зворотної сумісності розглядаються фрагментарно, без комплексних підходів до управління та стратегій її забезпечення;
- вартість підтримки API та витрати на міграцію споживачів майже не досліджуються, що свідчить про низький рівень досліджень економічних аспектів;
- бракує емпіричних досліджень, оскільки переважають теоретичні роботи, тоді як практичні кейси та кількісні дані представлені обмежено.

У контексті мікросервісної архітектури API відіграє роль системоутворювального елемента, що забезпечує комунікацію між сервісами шляхом передачі даних та команд, інтеграцію гетерогенних компонентів у єдину систему, абстракцію через приховування внутрішньої реалізації сервісів, а також контракт, що визначає чіткі правила взаємодії між компонентами. Без належним чином спроєктованих API мікросервіси стають ізольованими та не здатні утворити функціональну систему. Це підтверджується принципом автономності, який є одним із ключових принципів мікросервісної архітектури.

Для систематичного дослідження проблематики API доцільно застосувати методологічну рамку, що охоплює життєвий цикл API – від проєктування через версіювання і документування до моніторингу та застарівання, якісні атрибути API, зокрема продуктивність, надійність, безпеку, зрозумілість та зворотна сумісність, організаційні аспекти, включаючи управління змінами, комунікацію між командами та керівництвом, а також економічні фактори – кошторис розробки, підтримки та міграції споживачів. Такий комплексний підхід дозволяє охопити всі аспекти життєвого циклу API та ідентифікувати проблемні зони, що потребують наукового дослідження.

Однією з найгостріших проблем в управлінні мікросервісними системами є проблема «breaking changes» – змін в API, що порушують сумісність із попередніми версіями. Як зазначається в дослідженні Yaroshynskyi та інших [5], центральним питанням є виклики та ризики, пов'язані з модифікацією API під час еволюції програмного забезпечення, які можуть призвести до несподіваних наслідків. Зміни в API одного мікросервісу можуть спричинити каскадні відмови в усій системі. Collina та інші [3] безпосередньо адресують цю проблему, пропонуючи алгоритм оцінки ризику змін у мікросервісній архітектурі, при цьому автори наголошують, що кожна модифікація сервісу може спричинити відмову всієї системи, що робить управління ризиками критично важливим.

У літературі виділяють кілька підходів до версіювання API. URL-based версіювання передбачає включення версії в URL, наприклад /api/v1/resource. Header-based версіювання передбачає передачу версії в заголовках HTTP. Media

type версіювання ґрунтується на використанні Content-Type для визначення версії. Кожен із цих підходів має переваги та недоліки, однак систематичні дослідження їх ефективності в різних контекстах залишаються обмеженими.

Проблема крихкості мікросервісних систем тісно пов'язана з управлінням API. Система стає крихкою, коли зміни в одному сервісі непередбачувано впливають на інші, відсутні механізми виявлення та ізоляції відмов, документація API є застарілою або неповною, а також коли відсутні чіткі контракти між сервісами.

Висновки

Проведене дослідження дозволяє сформулювати такі висновки. По-перше, існує системний дефіцит уваги наукової спільноти до проблематики API в контексті мікросервісної архітектури, при цьому кількість публікацій з цієї теми є неспівмірно малою порівняно з загальним обсягом досліджень мікросервісів. По-друге, наявні публікації переважно констатують проблеми, але не пропонують системних методологій їх вирішення, особливо це стосується питань еволюції API та зворотної сумісності. По-третє, API є критичним елементом мікросервісних систем, що вимагає такого ж рівня наукової уваги, як і інші аспекти архітектури. Економічні аспекти, зокрема кошторис підтримки API та витрати на міграцію споживачів, залишаються майже недослідженими.

На підставі виявлених прогалин пропонуються такі напрямки майбутніх досліджень:

- розробка методологій сталого розвитку API, що передбачає системні підходи до управління еволюцією API з урахуванням потреб усіх стейкхолдерів;
- дослідження механізмів зворотної сумісності та аналіз ефективності різних підходів до її забезпечення;
- емпіричні дослідження вартості, спрямовані на кількісну оцінку витрат на підтримку API та міграцію споживачів у промислових умовах;
- дослідження автоматизації виявлення важливих змін, що передбачає розробку інструментів та методів для раннього виявлення потенційних проблем сумісності;
- стандартизація підходів та формування галузевих стандартів і найкращих практик управління API в мікросервісних системах;
- дослідження людських та організаційних факторів, зокрема вплив організаційної культури та процесів на ефективність управління API.

ЛІТЕРАТУРА / REFERENCE

1. Stojanov Z., Hristoski I., Stojanov J., Stojanov A. Research Trends and Recommendations for Future Microservices Research. Trudy ISP RAN/Proc. ISP RAS. 2024. Vol. 36, issue 1. P. 105–130. DOI: 10.15514/ISPRAS-2024-36(1)-7.
2. Lercher A. Managing API Evolution in Microservice Architecture. Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering. 2024.
3. Collina M., Maraschi L., Pirini T. Evaluating the Risk of Changes in a Microservices Architecture. arXiv:2309.06238. 2023.
4. Lahtinen S. Light-weight Method for Detecting API Breakages in Microservice Architectures. University of Helsinki. 2022.
5. Yaroshynskyi M., Puchko I., Prymushenko A., Kravtsov H. Investigating the Evolution of Resilient Microservice Architectures: A Compatibility-Driven Version Orchestration Approach. MDPI Digital. 2025.

LACK OF ATTENTION OF THE SCIENTIFIC COMMUNITY TO API PROBLEMS IN THE ERA OF MICROSERVICE ARCHITECTURE

Bohdan Kushnir, Oleksandr Ivanov

Abstract. *This study explores an anomaly in contemporary scientific discourse regarding microservice architecture: the systematic undervaluation of the role of Application Programming Interfaces (APIs) amidst a dominance of publications on cloud computing, artificial intelligence, and other emerging fields. An analysis of publications in leading scientometric databases for the period 2020–2025 reveals a significant imbalance: the volume of research on microservice architecture and related technologies substantially exceeds the number of works dedicated specifically to the management, evolution, and design of software interfaces. The API serves as a pivotal element of microservices; without it, components remain isolated and the system non-functional. The research identifies gaps in the scientific discourse, particularly the insufficient development of methodologies for sustainable API development, issues of backward compatibility, and the management of breaking changes.*

Keywords: *software, microservice architecture, API evolution, backward compatibility.*