МОДЕЛЮВАННЯ СЕРВІСНО-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Сірий Н.С., Турукало А.В..
Київський столичний університет імені Бориса Грінченка, м. Київ

ВСТУП

Актуальність і постановка проблеми. Стрімкий розвиток цифрових технологій та масова інтернетизація кардинально трансформували способи здійснення торгівлі. Електронна комерція перетворилася з нішевого каналу збуту на домінуючу модель роздрібного продажу: за даними Statista, обсяг світового ринку e-commerce у 2023 році перевищив 5,8 трлн доларів США, а до 2027 року прогнозується зростання до 7,9 трлн доларів [1]. Україна повторює загальносвітову тенденцію - попри виклики воєнного часу вітчизняний ринок онлайн-торгівлі залишається стійким, а потреба в надійних, масштабованих та гнучких програмних рішеннях для інтернет-магазинів не зменшується.

Ключовою технічною проблемою розробки сучасних комерційних платформ є вибір архітектурного підходу. Традиційна монолітна архітектура, де весь функціонал системи розгортається як єдиний неподільний артефакт, не задовольняє вимог сучасного бізнесу: будь-яка зміна вимагає повного перерозгортання, масштабування можливе лише цілком, а збій в одному модулі може паралізувати всю систему. Альтернативою є сервісно-орієнтована архітектура (SOA), що декомпозує систему на автономні сервіси з чітко визначеними інтерфейсами взаємодії. SOA дозволяє незалежно розробляти, масштабувати та оновлювати кожен функціональний модуль без впливу на решту системи. За даними Gartner, понад 70% нових e-commerce проектів 2023–2024 рр. реалізовано за headless/SOA-підходом [2], що підтверджує актуальність обраного напряму дослідження.

Мета дослідження полягає у проектуванні та програмній реалізації інформаційної системи інтернет-магазину на засадах сервісно-орієнтованої архітектури з використанням технологічного стеку React -Node.js/Express.js - PostgreSQL.

Аналіз останніх досліджень і публікацій. Проблематика архітектурного проектування систем електронної комерції є предметом активного наукового дискурсу. Фундаментальні принципи сервісно-орієнтованої архітектури систематизовано у монографії Т. Ерла [3], де визначено ключові характеристики сервісів: слабка зв'язаність, абстракція, повторне використання та автономність. М. Фаулер у праці «Patterns of Enterprise Application Architecture» [4] розглядає патерни проектування масштабованих систем – зокрема Repository та Unit of Work, - що широко застосовуються у SOA-рішеннях.

К. Річардсон [5] та С. Ньюмен [6] детально аналізують перехід від SOA до мікросервісної архітектури й зазначають, що SOA залишається оптимальним вибором для систем середнього масштабу, де складність мікросервісної інфраструктури (оркестрація контейнерів, розподілені трейси, управління конфігурацією) не є виправданою з огляду на розмір команди та бюджет проекту. За даними Stack Overflow Developer Survey 2023, JavaScript є найпопулярнішою мовою програмування одинадцятий рік поспіль, React та Node.js входять до трійки найзатребуваніших фронтенд- і бекенд-технологій відповідно [7]. Архітектурний стиль REST, формалізований у дисертаційній роботі Р. Філдінга [8], є де-факто стандартом побудови API у SOA-системах завдяки своїй простоті, масштабованості та підтримці у будь-якому сучасному середовищі розробки.

Короткий опис дослідження, його методів і засобів

Дослідження виконано із застосуванням комплексу теоретичних та практичних методів. На першому етапі проведено системний аналіз предметної галузі електронної комерції: визначено типові функціональні підсистеми інтернет-магазину (каталог, замовлення, оплата, автентифікація), вимоги до доступності (99,9% uptime), пропускної здатності та захисту персональних даних. Порівняльний аналіз архітектурних підходів та існуючих платформ (Shopify, WooCommerce, Magento) дозволив обгрунтувати вибір SOA як оптимального рішення для системи середнього масштабу.

На другому етапі застосовано об'єктно-орієнтоване та компонентне проектування для визначення складу і меж відповідальності сервісів, побудови схем бази даних та специфікації REST API. Межі сервісів визначалися за принципом bounded context – кожен сервіс інкапсулює окремий бізнес-домен і не розкриває деталей внутрішньої реалізації. Для моделювання взаємодії між компонентами системи використано діаграми послідовностей та компонентів, що відображають залежності між CatalogService, OrderService, AuthService і PaymentService.

Практичну реалізацію здійснено із дотриманням принципів чистої архітектури та SOLID. Кожен бекенд-сервіс організовано за трирівневою структурою: Router → Controller → Service → Repository, що забезпечує незалежне тестування кожного шару та спрощує заміну реалізації без зміни інтерфейсів.

Засоби розробки: React 18 [9] - бібліотека для побудови компонентного інтерфейсу з підтримкою хуків (useState, useEffect, useReducer, useContext) та React Router v6 для клієнтської маршрутизації; Node.js v20 LTS та Express.js [10] - серверна платформа і фреймворк для побудови REST API з підтримкою middleware-ланцюжків (валідація, автентифікація, логування); PostgreSQL 16 - реляційна СУБД з підтримкою JSONB-типів та GIN-індексів; Sequelize v6 - ORM для Node.js з

автоматизованим управлінням міграціями; JSON Web Token (RFC 7519) - стандарт безстанової автентифікації з access-токеном (24 год) та refresh-токеном (30 днів).

Функціональне тестування охопило перевірку всіх ключових сценаріїв: реєстрація та автентифікація, перегляд і пошук товарів, управління кошиком, оформлення та відстеження замовлення, адміністрування. Навантажувальне тестування виконано за допомогою Artillery з поступовим нарощуванням навантаження від 10 до 500 RPS. За результатами оптимізовано пул з'єднань Sequelize (max: 10, min: 2) та додано складові B-tree-індекси PostgreSQL для прискорення фільтрації товарів за категорією і ціновим діапазоном.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У ході порівняльного аналізу архітектурних підходів встановлено, що для системи електронної комерції середнього масштабу SOA є найбільш збалансованим рішенням. Детальне порівняння трьох підходів за ключовими критеріями наведено у таблиці 1.

Таблиця 1

Порівняльний аналіз архітектурних підходів до побудови e-commerce систем

Критерій	**Монолітна**	**SOA**	**Мікросервісна**
Масштабованість	Лише цілком	Сервісна	Незалежна
Ізоляція збоїв	Відсутня	Часткова	Повна
Складність розгорт.	Низька	Середня	Висока
Швидкість розробки	Висока (старт)	Середня	Низька (старт)
Технол. гнучкість	Низька	Середня	Висока
Оптимальний масштаб	Малий	Середній	Великий

Як видно з таблиці 1, SOA забезпечує сервісну масштабованість та часткову ізоляцію збоїв за помірної складності розгортання. Ці характеристики є оптимальними для інтернет-магазину, де чітко виділяються функціональні домени: каталог товарів, замовлення, автентифікація та платежі.

На основі аналізу предметної галузі спроектовано архітектуру системи, що складається з чотирьох незалежних бекенд-сервісів. CatalogService відповідає за управління товарами, категоріями та пошуком з підтримкою GIN-індексів PostgreSQL для повнотекстового пошуку. OrderService реалізує логіку обробки замовлень: створення, зміну статусу та відстеження доставки. AuthService забезпечує реєстрацію,

автентифікацію та авторизацію користувачів із використанням JWT-токенів з часом дії 24 години та механізмом оновлення (refresh token). PaymentService імітує взаємодію з платіжним шлюзом та обробляє транзакції з відповідними статусами. Кожен сервіс має власну схему бази даних і взаємодіє з іншими виключно через REST API.

Клієнтська SPA-частина на React 18 реалізує повний цикл взаємодії покупця: перегляд каталогу з мультикритеріальною фільтрацією та повнотекстовим пошуком, управління кошиком, оформлення замовлення з вибором адреси доставки, особистий кабінет з історією замовлень та адміністративну панель для управління товарами, замовленнями і користувачами. Маршрутизацію реалізовано засобами React Router v6, управління глобальним станом - через Context API з хуками useReducer та useContext. Захищені маршрути перевіряють наявність JWT-токена та роль користувача перед рендерингом компонента.

За результатами функціонального та навантажувального тестування підтверджено коректність реалізації всіх ключових сценаріїв використання системи. Результати навантажувального тестування основних ендпоінтів наведено у таблиці 2.

Таблиця 2

Результати навантажувального тестування основних ендпоінтів системи

Ендпоінт / Сервіс	**Середній час відповіді**	**Макс. RPS (без деградації)**
GET /api/v1/products (каталог)	38 мс	420 RPS
GET /api/v1/products/:id	12 мс	850 RPS
POST /api/v1/orders (замовлення)	74 мс	210 RPS
POST /api/v1/auth/login	45 мс	380 RPS
POST /api/v1/payments/process	820 мс	95 RPS

Аналіз результатів, наведених у таблиці 2, показав, що вузьким місцем є PaymentService - через імітацію затримки зовнішнього платіжного шлюзу (800 мс). Для решти сервісів середній час відповіді не перевищує 75 мс, що відповідає вимогам до систем електронної комерції щодо часу відгуку (до 200 мс). Оптимізація включала налаштування пулу з'єднань Sequelize (max: 10, min: 2, idle: 30 000 мс) та додавання складових індексів PostgreSQL для найчастіших запитів фільтрації товарів за категорією та ціновим діапазоном.

Схему бази даних побудовано навколо чотирьох основних сутностей. Сутність users зберігає облікові дані та профіль користувача; поле role

(enum: customer, admin) визначає рівень доступу. Сутність products містить атрибути товару, зокрема поля name та description з GIN-індексами для повнотекстового пошуку, а також stock_quantity для контролю залишків. Сутність orders пов'язана з users зовнішнім ключем і містить поле статусу (pending, confirmed, shipped, delivered, cancelled). Сутність order_items є проміжною таблицею між orders та products і фіксує кількість та ціну товару на момент замовлення, що виключає вплив майбутніх змін цін на історичні дані.

Специфікацію REST API розроблено відповідно до принципів RESTful-дизайну Р. Філдінга [8] з версіонуванням ендпоінтів (/api/v1/...) та дотриманням семантики HTTP-методів: GET - для отримання даних; POST - для створення ресурсів; PUT/PATCH - для повного та часткового оновлення; DELETE – для видалення. Захищені ендпоінти перевіряють JWT-токен у заголовку Authorization: Bearer <token> та роль користувача. Відповіді уніфіковано у форматі JSON з полями data, error та meta (пагінація), що спрощує обробку на клієнтській стороні та полегшує інтеграцію майбутніх клієнтів.

Порівняльний аналіз із готовими платформами (Shopify, WooCommerce, Magento) показав, що розроблена система вирізняється повною технологічною гнучкістю, відсутністю ліцензійних обмежень та можливістю точного налаштування бізнес-логіки без залежності від сторонніх хмарних провайдерів. Реалізований застосунок готовий до реального розгортання та може слугувати навчальним ресурсом для студентів спеціальностей «Програмна інженерія» і «Комп'ютерна інженерія».

ВИСНОВКИ

У ході дослідження встановлено, що сервісно-орієнтована архітектура є ефективним підходом до побудови масштабованих систем електронної комерції середнього рівня складності. На відміну від монолітної архітектури, SOA забезпечує незалежне масштабування окремих сервісів, ізоляцію збоїв та гнучкість оновлення компонентів без зупинки всієї системи. Порівняно з мікросервісною архітектурою, SOA не потребує складної інфраструктури оркестрації, що робить її доцільнішою для команд середнього розміру.

Розроблений застосунок на базі стеку React - Node.js - PostgreSQL підтвердив практичну ефективність обраного підходу: чотири незалежні сервіси успішно пройшли функціональне та навантажувальне тестування, а середній час відповіді не перевищив 75 мс для операцій читання. Результати тестування дозволили виявити та усунути вузькі місця у шарі доступу до даних.

Перспективами подальших досліджень є міграція до повноцінної мікросервісної архітектури з контейнеризацією (Docker/Kubernetes), впровадження асинхронної комунікації між сервісами через черги

повідомлень (RabbitMQ/Kafka), інтеграція реального платіжного шлюзу (Stripe/LiqPay), а також реалізація модуля персоналізованих рекомендацій на основі методів машинного навчання.

ДЖЕРЕЛА

1. Statista. E-commerce worldwide – Statistics & Facts [Електронний ресурс]. 2024. URL: https://www.statista.com/topics/871/online-shopping/ (дата звернення: 01.05.2026).
2. Gartner. Market Guide for Headless Commerce [Електронний ресурс]. 2023. URL: https://www.gartner.com/en/documents/4005918 (дата звернення: 01.05.2026).
3. Erl T. SOA: Principles of Service Design. Boston : Prentice Hall, 2007. 608 p.
4. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p.
5. Richardson C. Microservices Patterns: With Examples in Java. Shelter Island : Manning Publications, 2018. 520 p.
6. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 616 p.
7. Stack Overflow. Developer Survey 2023 [Електронний ресурс]. 2023. URL: https://survey.stackoverflow.co/2023/ (дата звернення: 01.05.2026).
8. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : дис. ... д-ра філос. / Університет Каліфорнії. Ірвін, 2000. 180 p.
9. Meta Open Source. React – A JavaScript library for building user interfaces [Електронний ресурс]. 2024. URL: https://react.dev/learn (дата звернення: 01.05.2026).
10. OpenJS Foundation. Node.js v20 LTS Documentation [Електронний ресурс]. 2024. URL: https://nodejs.org/en/docs (дата звернення: 01.05.2026).
11. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km