УДК 004.7:004.4

В. О. Бондар

ІНТЕГРАЦІЯ МОБІЛЬНИХ І ВЕБЗАСТОСУНКІВ ЧЕРЕЗ ЄДИНУ СЕРВЕРНУ СИСТЕМУ

Вінницький національний технічний університет

Анотація. *Розглянуто підхід до інтеграції мобільних і вебзастосунків через єдину серверну систему. Запропоновано архітектуру, у якій клієнтські застосунки використовують спільний програмний інтерфейс для доступу до бізнес-логіки, даних, засобів автентифікації та зовнішніх сервісів. Визначено основні переваги такого підходу, зокрема узгодженість даних, повторне використання функціональності, спрощення супроводу та централізований захист. Проаналізовано особливості синхронізації, версіонування API, роботи в автономному режимі й масштабування серверної частини.*

Ключові слова: інтеграція IT-систем, мобільний застосунок, вебзастосунок, серверна система, REST API, синхронізація даних, автентифікація.

Abstract. *The approach to integrating mobile and web applications through a unified backend system is considered. An architecture is proposed in which client applications use a common application programming interface to access business logic, data, authentication mechanisms, and external services. The main advantages of this approach are identified, including data consistency, functionality reuse, simplified maintenance, and centralized security. The features of synchronization, API versioning, offline operation, and backend scalability are analyzed.*

Keywords: IT systems integration, mobile application, web application, backend system, REST API, data synchronization, authentication.

Актуальність і постановка задачі

Сучасні інформаційні системи часто повинні одночасно підтримувати вебінтерфейс і мобільний застосунок. Реалізація для кожної платформи окремого сервера призводить до дублювання бізнес-логіки, відмінностей у правилах обробки даних та збільшення витрат на супровід. Доцільним рішенням є використання єдиної серверної системи, яка надає стандартизований програмний інтерфейс усім клієнтам. HTTP визначає уніфіковані правила обміну повідомленнями в розподілених системах, тому є базовим транспортним механізмом для такої інтеграції [1].

Метою роботи є визначення принципів побудови спільної серверної платформи для мобільного й вебзастосунків та аналіз архітектурних рішень, які забезпечують узгодженість даних, безпеку, масштабованість і незалежний розвиток клієнтських частин.

Архітектура інтегрованої системи

Єдина серверна система виконує роль централізованого рівня доступу до даних і функцій предметної області. Вебзастосунок та мобільний клієнт надсилають HTTPS-запити до REST API, отримуючи відповіді переважно у форматі JSON. Сервер перевіряє права доступу, виконує бізнес-операції, взаємодіє з базою даних, кешем і зовнішніми сервісами. Для формального опису контрактів доцільно застосовувати OpenAPI, що дає змогу автоматично створювати документацію, клієнтські бібліотеки та тести [2].

Узагальнену структуру інтеграції наведено на рисунку 1. Обидва клієнти використовують спільні моделі даних і правила предметної області, проте можуть мати різні механізми представлення, кешування та адаптації відповідей.

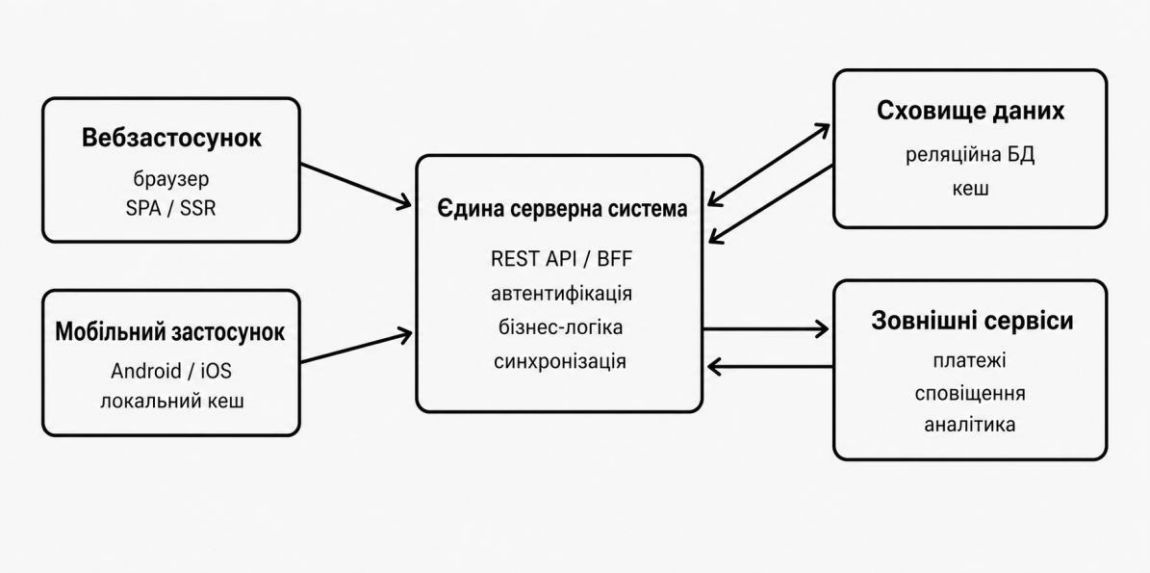

Рисунок 1 - Архітектура інтеграції мобільного і вебзастосунків

За збільшення кількості функцій сервер може бути побудований як модульний моноліт або сукупність мікросервісів. Перед сервісами доцільно розміщувати API Gateway, який виконує маршрутизацію, обмеження частоти запитів, журналювання та перевірку маркерів доступу. Якщо вимоги мобільного й вебклієнта суттєво відрізняються, застосовується шаблон Backend for Frontend: окремі адаптаційні шлюзи формують оптимізовані відповіді, але використовують спільні внутрішні сервіси [3].

Безпека та керування доступом

Централізована серверна система дає змогу однаково застосовувати політики автентифікації й авторизації до всіх клієнтів. Для делегованого доступу може використовуватися OAuth 2.0, який визначає механізм надання застосункам обмеженого доступу до HTTP-сервісів [4]. Передавання маркерів повинно здійснюватися лише через захищений канал HTTPS, а мобільний клієнт має зберігати секретні дані у захищеному сховищі операційної системи. Сучасні рекомендації OAuth також передбачають відмову від застарілих і менш безпечних потоків авторизації [5].

Сервер повинен реалізовувати перевірку вхідних даних, рольову або атрибутну модель доступу, журналювання критичних дій, обмеження частоти запитів і захист від повторного використання маркерів. Оскільки вебклієнт працює у браузері, окремо враховуються політика CORS, захист від CSRF і безпечне використання cookie. Для мобільного клієнта важливими є перевірка сертифікатів, контроль версії застосунку та мінімізація даних, що зберігаються локально.

Синхронізація даних і робота без мережі

На відміну від вебзастосунку, мобільний клієнт може тривалий час працювати за нестабільного з’єднання. Тому серверний API має підтримувати інкрементальне отримання змін, часові мітки або версії об’єктів. Локальні операції накопичуються в черзі та передаються після відновлення зв’язку. Конфлікти можуть вирішуватися за правилом останньої зміни, пріоритетом сервера або шляхом об’єднання полів залежно від предметної області.

Для запобігання повторному виконанню операцій запису до запиту додається ідемпотентний ключ. Сервер зберігає результат першої операції та повертає його у разі повторного запиту. Це особливо важливо для платежів, оформлення замовлень і створення інших критичних сутностей.

Порівняння основних підходів

Порівняння варіантів організації серверної взаємодії наведено в таблиці 1.

Таблиця 1 - Порівняння архітектурних підходів

Підхід	Переваги	Недоліки	Доцільність
Окремий сервер для кожного клієнта	Незалежність команд і релізів	Дублювання логіки та даних	Спеціалізовані, ізольовані продукти
Єдиний REST API	Спільна логіка, простіший супровід, узгоджені дані	Потрібно враховувати різні потреби клієнтів	Більшість малих і середніх систем
API Gateway та BFF	Оптимізація відповідей, незалежна адаптація клієнтів	Додаткові компоненти й складність	Великі системи з різними каналами доступу

Надійність, масштабування та супровід

Єдиний сервер є критичною точкою системи, тому його розгортання має передбачати резервування, балансування навантаження, моніторинг і автоматичне відновлення. Горизонтальне масштабування спрощується, якщо серверні екземпляри не зберігають користувацький стан у пам'яті, а сесії, кеш і файли розміщуються у спільних зовнішніх сховищах. Для зменшення затримок застосовуються кешування, стиснення відповідей, пагінація та вибіркове отримання полів.

Незалежне оновлення мобільного клієнта ускладнюється тим, що користувачі не завжди встановлюють нову версію одразу. Тому API необхідно версіонувати та протягом визначеного періоду підтримувати сумісність. Зміни контрактів повинні бути переважно розширювальними: додавання необов'язкових полів є безпечнішим, ніж перейменування або видалення існуючих. Автоматизовані контрактні та інтеграційні тести допомагають виявляти несумісність до розгортання.

Висновки

Інтеграція мобільних і вебзастосунків через єдину серверну систему забезпечує централізоване керування даними, бізнес-логікою та безпекою. Спільний API зменшує дублювання програмного коду й полегшує розвиток функціональності на різних платформах. Для практичної реалізації доцільно поєднувати REST API, формалізований контракт OpenAPI, централізовану автентифікацію та механізми синхронізації. У великих системах єдина точка входу може доповнюватися API Gateway і окремими BFF-компонентами. Надійність рішення визначається не лише вибором технологій, а й версіонуванням контрактів, автоматизованим тестуванням, моніторингом та продуманою обробкою автономної роботи мобільного клієнта.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Fielding R., Nottingham M., Reschke J. HTTP Semantics. RFC 9110. Internet Engineering Task Force, 2022. URL: https://www.rfc-editor.org/rfc/rfc9110.html (дата звернення: 07.06.2026).
2. OpenAPI Initiative. OpenAPI Specification. Version 3.2.0. 2025. URL: https://spec.openapis.org/oas/v3.2.0.html (дата звернення: 07.06.2026).
3. Microsoft. Backends for Frontends Pattern. Azure Architecture Center. 2025. URL: https://learn.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends (дата звернення: 07.06.2026).
4. Hardt D. The OAuth 2.0 Authorization Framework. RFC 6749. Internet Engineering Task Force, 2012. URL: https://www.rfc-editor.org/rfc/rfc6749.html (дата звернення: 07.06.2026).
5. Lodderstedt T., Bradley J., Labunets A., Fett D. Best Current Practice for OAuth 2.0 Security. RFC 9700. Internet Engineering Task Force, 2025. URL: https://www.rfc-editor.org/rfc/rfc9700.html (дата звернення: 07.06.2026).

Бондар Владислав Олександрович - студент групи 1ПІ-25м факультету інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: vladbondart11@gmail.com.

Науковий керівник: ***Ліщинська Людмила Броніславівна*** - доктор технічних наук, професор, Вінницький національний технічний університет, Вінниця.

Vladyslav Oleksandrovych Bondar - student, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: vladbondart11@gmail.com.

Scientific supervisor: ***Liudmyla Bronislavivna Lishchynska*** - Doctor of Technical Sciences, Professor, Vinnytsia National Technical University, Vinnytsia.