МОДУЛЬНА СИСТЕМА ВЕБ-СЕРВІСУ З МОЖЛИВІСТЮ РОЗШИРЕННЯ ПІД БІЗНЕС-МОДЕЛІ

Баранник Б. К., Вембер В.П.
Київський столичний університет імені Бориса Грінченка, м. Київ

ВСТУП

Стрімкий розвиток електронної комерції та цифрових сервісів зумовлює необхідність створення гнучких, масштабованих і безпечних веб-платформ. Сучасні маркетплейси забезпечують централізовану взаємодію між продавцями, покупцями та адміністраторами, підтримують розміщення оголошень, обробку замовлень і фінансові розрахунки. Водночас більшість існуючих рішень розробляються як вузькоспеціалізовані продукти, що ускладнює їх подальший розвиток і адаптацію під нові бізнес-вимоги.

Актуальність і постановка проблеми. Актуальною проблемою є забезпечення масштабованості, розширюваності та безпеки програмних систем. Зростання кількості користувачів, ролей, типів контенту та бізнес-процесів вимагає застосування модульних архітектурних підходів. Традиційні підходи, що покладаються виключно на перевірки на рівні серверної логіки, не завжди забезпечують достатній рівень захисту. Перспективним є використання механізмів безпеки на рівні бази даних — зокрема Row Level Security (RLS) у PostgreSQL, — які обмежують доступ до записів залежно від ролі та контексту користувача.

Мета дослідження. Спроєктувати та реалізувати модульну систему веб-сервісу маркетплейсу з адміністративною панеллю, що забезпечує масштабованість, розширюваність і захист даних на рівні бази даних, з перспективою адаптації під різні бізнес-моделі у сфері електронної комерції.

Для досягнення мети вирішено такі завдання: проаналізовано предметну область і існуючі програмні рішення; спроєктовано архітектуру системи, структуру бази даних і моделі доступу; розроблено MVP-версію веб-сервісу з серверною та клієнтською частинами, функціоналом управління оголошеннями, адміністративною панеллю та інтеграцією платіжної системи; забезпечено захист даних і проведено тестування.

Аналіз останніх досліджень і публікацій. Питання проєктування веб-сервісів досліджуються у працях провідних спеціалістів з програмної інженерії. Richards M. та Ford N. [1] розглядають фундаментальні принципи програмних архітектур і обґрунтовують доцільність модульного підходу. Newman S. [2] детально описує підходи до побудови мікросервісів і принципи слабкого зв’язування компонентів. Bass L. та ін. [3] аналізують практичне застосування різних архітектурних стилів. Fielding R. T. [4] заклав концептуальні основи REST-архітектури. Проблеми безпеки на рівні бази даних у контексті багатокористувацьких веб-систем

розглядаються у документації PostgreSQL [5] і матеріалах OWASP [6]. Разом із тим, питання поєднання модульного монорепозиторного підходу з RLS-безпекою на рівні BaaS-платформ у контексті маркетплейсів потребують подальшого дослідження.

Короткий опис дослідження, його методів і засобів

У роботі використано такі методи дослідження: аналіз предметної області та існуючих програмних рішень у сфері маркетплейсів; системний аналіз архітектур веб-систем; об'єктно-орієнтований та компонентний підходи до розробки програмного забезпечення; проєктування баз даних та моделей доступу із застосуванням рольової моделі і принципу мінімально необхідних прав; практичне програмування з використанням сучасних веб-технологій; функціональне та інтеграційне тестування системи.

Технологічний стек системи сформовано з урахуванням вимог до продуктивності, типобезпеки та підтримуваності. Клієнтська частина: React (компонентний підхід, SPA) + TypeScript (статична типізація) + Vite (швидка збірка) + Tailwind CSS (утилітарне стилювання) + React Router (маршрутизація) + TanStack Query (управління асинхронними запитами і кешуванням). Серверна частина: Node.js + Express.js — забезпечує реалізацію REST API, перевірку прав доступу та бізнес-логіку. Використання TypeScript на сервері разом із клієнтом забезпечує узгодженість типів у межах монорепозиторію. Рівень даних і безпеки: Supabase (BaaS-платформа на базі PostgreSQL) — управління даними, автентифікація (Supabase Auth із токенами доступу), зберігання файлів (Supabase Storage з bucket-правилами), Row Level Security. Service Role Key використовується виключно на стороні сервера для адміністративних операцій.

Організація проєкту: монорепозиторій з pnpm workspaces — централізоване управління залежностями, узгодженість версій бібліотек, спільні конфігурації для frontend і backend. Каталог supabase містить SQL-скрипти створення схеми, RLS-політик та початкового наповнення. Для розгортання використано Vercel (клієнтська частина), Render (серверна частина) та Supabase (база даних і файлове сховище) із безкоштовними тарифними планами на початковому етапі.

Архітектура побудована за принципом модульного моноліту — підходу, що поєднує простоту монолітного застосунку з ізольованістю компонентів, характерною для мікросервісів. Кожен функціональний блок (автентифікація, оголошення, замовлення, адміністрування, платежі) реалізовано як окремий модуль з власними маршрутами, контролерами та рівнем доступу до даних. Взаємодія між клієнтом і сервером здійснюється через REST API по HTTPS. Ключовою особливістю є багаторівнева модель безпеки: клієнтський рівень (приховування елементів UI залежно від ролі) + серверний рівень (перевірка токенів і ролей перед кожною критичною операцією) + рівень бази даних (RLS-політики для всіх основних таблиць).

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У результаті дослідження спроєктовано та реалізовано повнофункціональний MVP веб-сервісу маркетплейсу. Нижче описано ключові компоненти та отримані результати.

1. Структура бази даних. Спроєктовано реляційну схему з п’яти основних таблиць: profiles (профілі та ролі користувачів, зв’язана з Supabase Auth через тригер), categories (категорії оголошень, керовані адміністратором), listings (оголошення з полями owner_id → profiles і category_id → categories, атрибутами опису, зображень та статусу), orders (замовлення з посиланнями buyer_id та listing_id, даними доставки та статусом), admin_audit_log (журнал адміністративних дій із admin_id → profiles). Між таблицями встановлено зв’язки «один-до-багатьох» із зовнішніми ключами та обмеженнями цілісності на рівні бази даних.

2. Рольова модель та контроль доступу (RLS). Реалізовано дві ролі: user та admin. Механізм Row Level Security визначає правила на рівні кожної таблиці. Для таблиці profiles: перегляд і редагування дозволено лише для власного запису; адміністратор має доступ до всіх профілів. Для таблиці listings: звичайний користувач бачить лише активні оголошення інших та всі власні; редагування і видалення — лише власнику або адміністратору; заблокований користувач позбавляється права на створення нових оголошень через відповідну умову в RLS-політиці. Для таблиці orders: покупець має доступ до власних замовлень, продавець — до замовлень, пов’язаних з його оголошеннями, адміністратор — до всіх. Таблиця admin_audit_log доступна виключно адміністраторам. Контроль доступу на рівні бази даних мінімізує ризики витоку даних навіть у разі помилок у прикладному коді.

3. Модуль автентифікації та управління користувачами. Реєстрація, вхід та управління сесіями реалізовано через Supabase Auth з токенами доступу. Після реєстрації автоматично формується запис у таблиці profiles через тригер бази даних із призначенням ролі user за замовчуванням. Серверна частина перевіряє токен у заголовку кожного запиту та визначає роль користувача перед виконанням будь-якої критичної операції. Механізм блокування реалізовано через поле is_blocked у таблиці profiles — заблокований обліковий запис обмежується як через перевірки серверної логіки, так і через RLS-умови.

4. Модуль оголошень. Забезпечено повний цикл CRUD-операцій: створення оголошення з вибором категорії, завантаженням зображень до Supabase Storage та зазначенням контактних даних; редагування та зміна статусу (active / archived / blocked); перегляд з фільтрацією за категоріями. Незареєстровані користувачі мають доступ до перегляду активних оголошень без автентифікації. Обмеження на кількість і тип зображень реалізовано на рівні серверної валідації та bucket-правил Supabase Storage.

5. Модуль замовлень. Покупець оформлює замовлення із зазначенням адреси доставки, контактного телефону та кількості. Система зберігає замовлення зі статусом та зв'язком з оголошенням і покупцем. Перегляд замовлень розмежовано: розділ «Мої замовлення» відображає покупки поточного користувача, «Мої продажі» — замовлення на його оголошення. Зміна статусу замовлень контролюється серверною частиною, що запобігає несанкціонованим маніпуляціям з боку клієнта.

6. Адміністративна панель. Реалізовано такі функції: CRUD-операції над категоріями; модерація оголошень — перегляд усіх оголошень незалежно від статусу та зміна статусу (active / archived / blocked); управління користувачами — перегляд облікових записів і блокування; журнал аудиту — автоматична фіксація дій адміністратора в таблиці admin_audit_log (зміни статусів, блокування, редагування категорій). Доступ до адміністративної панелі обмежено роллю admin на всіх рівнях: UI, серверна логіка, RLS. Усі адміністративні операції виконуються через Service Role Key виключно на сервері.

7. Інтеграція платіжної системи LiqPay (MVP). Вибір LiqPay обумовлено його поширеністю в Україні та наявністю sandbox-режиму для тестування без реальних фінансових операцій. Процес оплати ініціюється користувачем зі сторінки замовлення; сервер перевіряє коректність даних і формує платіжний запит, повертаючи параметри для переходу до платіжної форми. Взаємодія з платіжним сервісом повністю реалізована на сервері, що унеможливлює передачу конфіденційних ключів клієнтській частині. Підтвердження оплати обробляється умовно (MVP-режим) з відповідним оновленням статусу замовлення.

8. Захист конфігураційних параметрів. Усі чутливі дані (ключі Supabase, Service Role Key, параметри LiqPay) зберігаються у змінних середовища і не включаються до репозиторію. У репозиторії зберігається лише файл .env.example з переліком необхідних змінних без фактичних значень. Передача даних між клієнтом і сервером захищена протоколом HTTPS. Зберігання файлів у Supabase Storage організовано з bucket-правилами, що обмежують завантаження та перегляд залежно від ролі користувача.

9. Тестування. Проведено функціональне та інтеграційне тестування всіх основних модулів. Тестові сценарії охоплювали: реєстрацію та автентифікацію (включно зі створенням профілю через тригер); CRUD-операції з оголошеннями та перевірку видимості для різних ролей; оформлення замовлень і розмежування доступу між покупцем і продавцем; адміністративний функціонал і запис до журналу аудиту; коректність RLS-політик — підтверджено, що пряме виконання SQL-запитів не надає доступу до чужих даних. Результати тестування підтвердили відповідність системи визначеним функціональним вимогам і коректність реалізації механізмів безпеки.

ВИСНОВКИ

У ході дослідження спроєктовано та реалізовано модульну систему веб-сервісу маркетплейсу на основі React, TypeScript, Node.js, Express, PostgreSQL та Supabase. Ключовою перевагою розробленого рішення є реалізація безпеки на рівні бази даних за допомогою Row Level Security, яка гарантує захист даних незалежно від рівня прикладної логіки. Модульна архітектура на базі монорепозиторію забезпечує розширюваність системи та можливість її адаптації під різні бізнес-моделі без повної переробки. Реалізовано повний цикл взаємодії користувачів: від реєстрації та публікації оголошень до оформлення замовлень, модерації контенту та інтеграції платіжної системи LiqPay. Розроблений MVP може бути використаний як основа для створення повноцінної комерційної платформи електронної торгівлі.

ДЖЕРЕЛА

1. Richards M., Ford N. Fundamentals of Software Architecture. O'Reilly Media, 2020. 448 с.

2. Newman S. Building Microservices. O'Reilly Media, 2015. 280 с.

3. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Addison-Wesley, 2013. 624 с. дата звернення: «10.03.2026»

4. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: doctoral dissertation. University of California, Irvine, 2000. 180 с.

5. PostgreSQL Row Level Security [Електронний ресурс]. URL: https://www.postgresql.org/docs/current/ddl-rowsecurity.html дата звернення: «05.02.2026»

6. OWASP Top 10 Web Application Security Risks [Електронний ресурс]. URL: https://owasp.org/www-project-top-ten дата звернення: «23.04.2026»

7. Supabase Documentation [Електронний ресурс]. URL: https://supabase.com/docs дата звернення: «05.02.2026»

8. Хлиста, І. (2023). Вирішення проблеми часткового оновлення вмісту веб-сторінки шляхом оптимізації параметрів SPA. Комп'ютерне моделювання та інформаційні технології, с. 286-291.

9. React Documentation [Електронний ресурс]. URL: https://react.dev дата звернення: «05.02.2026»

10. 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