УДК 004.41
Радюк В. - ст. гр. СП-42
*Тернопільський національний технічний університет імені Івана Пулюя*

# АРХІТЕКТУРНІ ПІДХОДИ ДО ПРОЄКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ ВЕБ – ЗАСТОСУНКІВ

Науковий керівник: к.т.н., доцент Цуприк Г.Б.

Radiuk V.
*Ternopil Ivan Puluj National Technical University*

# ARCHITECTURAL APPROACHES TO DESIGNING THE BACKEND OF WEB APPLICATIONS

Supervisor: PhD, Associate Professor H. B. Tsupryk

Ключові слова: запити, мікросервіси, моноліт
Keywords: requests, microservices, monolith

Серверна частина веб-додатків відповідає за обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних. Вона забезпечує автентифікацію та авторизацію користувачів, обробку помилок, логування подій та інтеграцію із зовнішніми сервісами. Крім того, сервер обробляє HTTP-запити, виконує операції створення, читання, оновлення та видалення даних (CRUD) і повертає відповіді клієнту. При проєктуванні таких систем ключовим є вибір архітектурного підходу, який визначає масштабованість, підтримуваність та продуктивність застосунку [1].

Сучасні серверні застосунки також повинні відповідати вимогам високої доступності та відмовостійкості, що досягається шляхом використання балансування навантаження, резервування ресурсів і механізмів автоматичного відновлення. З огляду на зростання кількості користувачів і обсягів даних, особливого значення набуває оптимізація продуктивності, яка включає кешування результатів запитів, ефективну роботу з базами даних та мінімізацію затримок при обробці запитів. У цьому контексті широко застосовуються інструменти моніторингу, які дозволяють відстежувати стан системи в реальному часі та оперативно реагувати на можливі збої.

Найпоширенішими підходами є монолітна та мікросервісна архітектури. Монолітна модель передбачає реалізацію всієї функціональності в межах одного застосунку, що спрощує розробку та розгортання, але ускладнює масштабування окремих компонентів. Мікросервісна архітектура базується на розділенні системи на незалежні сервіси, кожен з яких відповідає за окрему функціональність, що забезпечує гнучкість і горизонтальне масштабування [2].

Крім того, у мікросервісних системах кожен сервіс може бути реалізований із використанням різних технологій і мов програмування, що дозволяє обирати найбільш ефективні інструменти для вирішення конкретних задач. Водночас це створює додаткові виклики, пов’язані з оркестрацією сервісів, забезпеченням їхньої взаємодії та управлінням конфігураціями. Для вирішення цих задач застосовуються контейнери та системи керування контейнерами, такі як Docker і Kubernetes.

При проєктуванні серверної частини використовуються принципи розділення відповідальностей (Separation of Concerns) та багаторівнева архітектура, яка включає рівні представлення, бізнес-логіки та доступу до даних. Застосування шаблонів, таких

як MVC або Clean Architecture, дозволяє підвищити модульність і спростити підтримку коду. Для взаємодії між клієнтом і сервером використовується REST-підхід, який визначає стандартизовану взаємодію через HTTP-протокол і є одним із ключових архітектурних стилів веб-систем [1].

Окрім REST, у сучасних системах також можуть використовуватися альтернативні підходи до обміну даними, зокрема GraphQL або gRPC, які забезпечують більшу гнучкість у формуванні запитів і підвищення ефективності передачі даних. Вибір конкретного підходу залежить від вимог до продуктивності, складності системи та характеру взаємодії між клієнтом і сервером.

Важливим аспектом є забезпечення цілісності даних та коректної роботи в умовах багатокористувацького доступу, що досягається за рахунок використання транзакцій і контролю конкурентного доступу. У мікросервісних системах ці задачі ускладнюються через розподіленість сервісів і потребують додаткових підходів до узгодженості даних [2].

Для забезпечення узгодженості в таких системах застосовуються підходи, як-от eventual consistency, шаблони Saga або використання брокерів повідомлень для асинхронної взаємодії між сервісами. Це дозволяє зменшити зв’язність між компонентами та підвищити стійкість системи до збоїв. Таким чином, ефективне проєктування серверної частини веб-додатків вимагає комплексного підходу, що враховує як архітектурні, так і технологічні аспекти побудови програмних систем.

Література:

1. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. Roy Fielding. URL: https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf.
2. Fowler M. Microservices. *Martin Fowler*. *Articles*. 25.03.2014. URL: https://martinfowler.com/articles/microservices.html.
3. Olianin D., Tsupryk H., Hovorushchenko T., Bahrii-Zaiats O., Andrushchak I. Transformer Neural Networks in Industry 4.0. Proceedings of the 3rd International Workshop (CITI-2025), Ternopil, Ukraine, 11–12 June 2025. Ternopil: Ternopil Ivan Puluj National Technical University, 2025. URL: https://ceur-ws.org/Vol-4057/
4. Tsupryk H., Olianin D. Data extraction from text using Transformer Neural Networks. Information Technology: Computer Science, Software Engineering and Cyber Security. 2025. №2. P. 125–130. URL: https://doi.org/10.32782/IT/2025-2-13
5. Olianin D., Tsupryk H. Огляд ролі трансформерних нейроних мереж у видобувані інформації із неструктурованих даних. Measuring and computing devices in technological processes. 2025. Vol. 82(2). P. 360–364. URL: https://doi.org/10.31891/2219-9365-2025-82-52