УДК 004.9:629.331.083.5

КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА В СИСТЕМІ ПІДТРИМКИ РОБОТИ МЕРЕЖІ ПІДПРИЄМСТВ АВТОСЕРВІСУ

Харченко А.І., Вишняк М.Ю.
e-mail: anna.kharchenko@nure.ua, mykhailo.vyshniak@nure.ua
Харківський національний університет радіоелектроніки, каф. СТ
м. Харків, Україна

The report explores the client-server architecture, structured into three distinct layers: the web client, the web server, and the relational database. The web client functions as a browser-based application, offering an intuitive user interface. The web server processes client requests, executes business logic, and communicates with the database. The database, designed as a relational system using InnoDB storage, eliminates the reliance on paper-based records and enhances data processing at the SQL server level by procedures, triggers, and indexes. The adoption of this architectural framework facilitates a substantial enhancement in productivity and optimization of resource management. These elements are essential for the dynamic evolution of the service station network.

При автоматизації бізнес-функцій для компанії, що володіє мережею автосервісів, використана клієнт-серверна архітектура інформаційної системі. Архітектура системи розділена на три ланки: рівень клієнта (Front-end), рівень застосунку (Back-end) та рівень зберігання даних (Database Layer) [1, 2].

Рівень клієнта представлений веб-застосунком, що працює у браузерах: Google Chrome, Mozilla Firefox, Safari та інші. Завданням даного рівня є взаємодія з веб-сервером через REST API та надання зручного користувацького інтерфейсу. При реалізації рівня клієнта використаний фреймворк React.js.

Рівень застосунку відповідає за обробку запитів клієнта, виконання бізнес-функцій та за комунікацію з базою даних. Серверна частина побудована з допомогою Java Spring Boot. Для розгортання серверного застосунку використовується Apache Tomcat, що є контейнером сервлетів і функціонує як веб-сервер.

Також на рівні застосунку реалізований контроль доступу та цілісність даних завдяки використанню фреймворку Spring Security. Це зменшує ризик SQL-ін'кцій та несанкціонованих запитів [3].

Програмний код серверної частини представлений у структурі Model-View-Controller для розділення класів за їх відповідальністю [4]. У даній структурі можна виділити блоки моделі, представлення та контролеру:

− блок моделі представляє класи, що містять бізнес-логіку. Наприклад, алгоритми створення та обробки замовлень на технічне обслуговування авто та керування графіками механіків;

− блок представлення містить класи, що визначають спосіб відображення даних для користувача, де був обраний JSON-формат даних

для роботи з REST API. Також цей рівень відповідає за генерацію веб-сторінок для користувача;

− блок контролеру є посередником між згаданими блоками. Класи-контролери відповідають за обробку HTTP-запитів, перевіряють дані і викликають методи класів-моделей та повертають користувачеві згенеровані представлення.

Реляційна база даних MySQL є третім рівнем клієнт-серверної архітектури. Вона позбавляє паперового обліку діяльності компанії і зберігає дані про користувачів та їх автомобілі, менеджерів компанії, перелік послуг з обслуговування авто та їх вартість, механіків та їх робочі графікі.

У базі даних MySQL використовуються таблиці на механізм InnoDB, який підтримує транзакції й обмеження по зовнішньому ключу та відповідає принципам ACID [5] і також використовуються: процедури для прискорення виконання складних операцій; тригери для автоматичного виконання дій при змінах в БД та індекси для зменшення часу пошуку в таблицях.

Взаємодія між рівнями застосунку та бази даних відбувається через Spring Data JPA та ORM-фреймворк Hibernate.

Використання триланкової клієнт-серверної архітектури дозволяє створити гнучку, масштабовану та надійну систему для автоматизації роботи мережі автосервісів. Чітке розділення системи дозволиляє ізолювати кожний компонента за сферою їх відповідальність.

При збільшенні навантаження на мережу підприємств автосервісу можна виконати горизонтальне масштабування системи завдяки даній архітектури клієнт-сервер.

Використання структури Model-View-Controller полегшує розробку, тестування та підтримку програмного коду. Це сприяє незалежному оновленню окремих блоків без порушення загальної роботи веб-серверу.

Список використаних джерел:

1. Елементи системного проектування : навч. посібник. / І.В. Гребеннік та ін. Харків: ХНУРЕ, 2016. 322 с.

2. Goodyear M. Enterprise System Architectures: Building Client Server and Web Based Systems. Boca Raton : CRC Press, 2017. 978 p.

3. Paraschiv E. Control the Session with Spring Security. Baeldung. URL: https://www.baeldung.com/spring-security-session (дата звернення: 03.03.2025).

4. Necula S. Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications. Preprints. 2024. DOI: 10.20944/preprints202404.1860.v1 (дата звернення: 03.03.2025).

5. Офіційна документація MySQL Workbench. URL: https://dev.mysql.com/doc/workbench/en/ (дата звернення: 03.03.2025).