УДК 004.7

Леус В. О., здобувач,
Савіцький Р. С., ст. викладач, аспірант
Державний університет «Житомирська політехніка»

МЕТОДИ СТРУКТУРИЗАЦІЇ ТА ЗБЕРЕЖЕННЯ ДАНИХ У СИСТЕМАХ ВІЗУАЛЬНОГО КОНСТРУЮВАННЯ ВЕБРЕСУРСІВ

Стрімкий розвиток платформ для створення вебресурсів сильно змінив підхід до розробки ПЗ. Сучасні редактори дозволяють користувачам без навичок програмування створювати складні інтерфейси за допомогою механізмів Drag&Drop. Проте для реалізації потрібно вирішити нетривіальну архітектурну задачу: як ефективно структурувати / зберігати створений динамічний контент, структура якого заздалегідь невідома та може змінюватися в будь-який момент.

Традиційний підхід, за яким контент зберігається як готова HTML-розмітка, є неприйнятним для NoCode систем. Збереження HTML у чистому вигляді призводить до втрати семантичного зв'язку елементів, що ускладнює редагування блоків та унеможливлює зміну глобального дизайну сайту надалі. Альтернативою є збереження структури інтерфейсу у вигляді серіалізованого дерева об’єктів (подібного до AST), де кожен вузол є окремим функціональним блоком (текст, зображення, форма) з власними властивостями.

Використання деревоподібних структур висуває специфічні вимоги до бази даних. Класичні реляційні бази даних (SQL) вимагають жорстко заданих схем таблиць. Спроба нормалізувати динамічне дерево блоків у реляційній моделі призводить до використання антипатерну Entity-Attribute-Value, що знижує продуктивність системи через необхідність виконувати велику кількість складних операцій JOIN під час завантаження однієї сторінки.

З іншого боку, документо-орієнтовані бази даних (NoSQL) ідеально підходять для збереження ієрархічних JSON-структур. Однак конструктори містять сутності зі строгою реляційною природою: облікові записи користувачів, права доступу, платежі, зв'язки між сайтами. Застосування виключно NoSQL підходу ускладнює забезпечення транзакційної цілісності для цих критичних даних.

Оптимальним архітектурним рішенням для NoCode платформ є використання гібридної моделі збереження даних на базі сучасних реляційних СУБД (MySQL версії 8.0+), які мають нативну підтримку типу даних JSON [1].

У межах такої моделі архітектура бази даних розділяється. Статичні сутності (користувачі, сайти) зберігаються у класичних реляційних таблицях, що гарантує цілісність даних та каскадне видалення. А динамічна структура сторінки (дерево блоків) зберігається в полі типу JSON. Кожен об'єкт у цьому масиві містить ідентифікатор типу компонента, його унікальний ключ, параметри стилізації та безпосередньо контент.

Переваги такого підходу для систем візуального конструювання є очевидними.

Однією з переваг є швидкодія читання, тому що вся конфігурація сторінки завантажується з бази даних одним запитом без об’єднань, незалежно від кількості блоків або рівня їх вкладеності.

Також гнучкість інтерфейсу дозволяє додавати нові типи блоків до редактора і не потребує міграції або змін схем бази даних на сервері, що значно прискорює подальшу розробку.

Підтримка цілісності системи дає можливість використовувати зовнішні ключі для з’єднання сторінок із сайтами та їх власниками.

Для клієнтського застосунку збережена JSON-структура є декларативною схемою [2]. Під час завантаження сторінки модуль рендерингу парсить JSON-дерево і рекурсивно перетворює кожен вузол у відповідний компонент, передаючи збережені властивості як параметри. Це відокремлює дані від логіки відображення [3].

Ефективне функціонування сучасних NoCode платформ неможливе без оптимізації зберігання даних. Відмова від збереження готового HTML на користь серіалізованих JSON-дерев об’єктів дозволяє реалізувати повноцінний компонентний підхід, а використання гібридної моделі зберігання в реляційній базі даних надає оптимальний баланс між гнучкістю, необхідною для збереження довільних структур інтерфейсу, та транзакційною цілісністю, що є критичним для управління даними користувачів та бізнес-логікою системи.

Список використаних джерел:

1. MySQL 8.0 Reference Manual. The JSON Data Type. URL: https://dev.mysql.com/doc/refman/8.0/en/json.html (дата звернення: 09.03.2026).

2. Brooks R. A Deep Dive into Airbnb’s Server-Driven UI System. Airbnb Engineering. URL: https://medium.com/airbnb-engineering/a-deep-dive-into-airbnbs-server-driven-ui-system-842244c5f5 (дата звернення: 11.03.2026).

3. Thinking in React. The Official React Documentation. URL: https://react.dev/learn/thinking-in-react (дата звернення: 11.03.2026).