УДК 004.8

Н. В. Кононенко
С. І. Петришин
С. В. Сімончук

РОЗРОБКА WEB-ДОДАТКУ ДЛЯ CMS З ВИКОРИСТАННЯМ HEADLESS ПІДХОДУ

Вінницький національний технічний університет

Анотація

Проведено аналіз переваг та недоліків використання Headless підходу порівняно з традиційними підходами розробки повнофункціональних WEB-додатків.

Ключові слова: Headless, CMS, Headless CMS, WordPress, блог, доставка контенту.

Abstract

An analysis of the advantages and disadvantages of using the Headless approach compared to traditional approaches for developing full-featured web applications has been conducted.

Keywords: Headless, CMS, Headless CMS, WordPress, blog, content delivery.

Вступ

Зростання вимог до сучасних WEB-додатків стимулює пошук нових архітектурних підходів до їх розробки. Традиційні Fullstack рішення, в яких бекенд та фронтенд реалізуються у єдиній монолітній системі, часто виявляються недостатньо гнучкими для масштабованих або мультиканальних рішень. Однією з актуальних альтернатив є використання Headless-підходу, що передбачає розділення клієнтської (frontend) та серверної (backend) частин, де CMS виконує роль централізованого сховища контенту та джерела даних [1].

У процесі дослідження було проаналізовано основні підходи до розробки WEB-додатків, особливості використання традиційних Fullstack-архітектур, а також сучасних концепцій на базі Headless CMS. Проведено порівняльний аналіз їхніх переваг і недоліків у контексті вимог до продуктивності, масштабованості, гнучкості інтеграцій та мультиканальності. За результатами аналізу було обґрунтовано доцільність використання Headless підходу для побудови універсального WEB-додатку на базі CMS з використанням Headless підходу.

Результати дослідження

Головна проблема традиційної архітектури WEB-додатків полягає у тому, що зі зростанням вимог до швидкодії та гнучкості інтерфейсів стає все важче забезпечити якісне користувацьке враження при використанні класичних монолітних рішень. Інтегрована структура, в якій серверна та клієнтська частини тісно пов’язані між собою, ускладнює масштабування та унеможливлює незалежний розвиток окремих компонентів.

Виникає логічне питання: чому не залишити все у межах однієї системи? Проблема полягає у тому, що для сучасних сценаріїв використання (наприклад, мобільні застосунки, PWA, SPA) потрібен гнучкий і незалежний доступ до контенту з різних каналів, що важко забезпечити за допомогою класичного пов’язаного підходу. Будь-які зміни у клієнтській частині тягнуть за собою зміни у серверній, що значно уповільнює розробку та підвищує ризик виникнення помилок.

Саме для вирішення цих проблем активно розвиваються сучасні підходи до архітектури WEB-додатків. Виділяють три основні підходи:

1. Fullstack-підхід – створення повного стека застосунку з нуля (backend + frontend), з використанням гнучких інструментів під конкретні завдання.

2. Традиційна CMS-архітектура – використання готових систем керування контентом із інтегрованим інтерфейсом (наприклад, WordPress з класичними темами).
3. Headless-підхід – відокремлення логіки зберігання та керування контентом (API-first CMS) від клієнтської частини, що дозволяє створювати легкі та високопродуктивні інтерфейси за допомогою сучасних фреймворків.

Fullstack підхід

Fullstack підхід базується на повному розділенні frontend і backend-частин. У цьому випадку frontend створюється як окремий застосунок із використанням сучасних JavaScript-фреймворків, а серверна частина – незалежно, на відповідній платформі. Обмін даними між цими компонентами здійснюється через REST API або GraphQL. Водночас впровадження такого підходу потребує значних зусиль на етапі налаштування інфраструктури і вимагає глибшої технічної компетенції. Це також збільшує тривалість розробки та пов'язані з нею витрати [2].

Цей підхід також передбачає розділення системи на кілька окремих компонентів: клієнтську частину (Client), WEB-сервер (WEB Server), сервер додатків (Application Server) та базу даних (Database). Клієнт надсилає запити до WEB-сервера, який відповідає за їх прийом і передачу серверу додатків для обробки бізнес-логіки. Сервер додатків взаємодіє з базою даних для збереження або отримання необхідної інформації. Після обробки дані повертаються через WEB-сервер до клієнта, що забезпечує розподіл функцій, гнучкість і масштабованість системи.

Класична CMS

Передбачає використання вбудованого шаблонізатора в межах CMS. У цьому випадку WEB-додаток створюється безпосередньо у середовищі системи керування контентом. Розмітка сторінок, обробка даних та виведення контенту реалізуються за допомогою вбудованих шаблонів та механізмів CMS. Прикладами такого підходу є сайти на базі WordPress із використанням PHP-шаблонів, Joomla або Drupal із системою Twig-шаблонів [3].

Такий підхід дозволяє швидко запускати проєкти невеликого та середнього масштабу, використовуючи численні готові шаблони і плагіни. Однак при спробі розширити функціонал або інтегрувати сучасний динамічний інтерфейс, ці системи часто виявляються обмеженими. Крім того, масштабування додатків у цій архітектурі зазвичай призводить до зростання складності та зниження продуктивності, особливо при великому навантаженні.

Headless підхід

Найбільш сучасним і перспективним підходом у контексті розробки складних WEB-додатків є використання Headless-архітектури. У такій моделі CMS або інша система управління контентом функціонує виключно як backend-сервіс для зберігання та керування даними, які передаються через API. Логіка відображення інформації та взаємодії з користувачем реалізується у frontend-додатку окремо за допомогою спеціалізованих фреймворків. Це дозволяє досягти мультиплатформності: один і той самий API може обслуговувати як WEB-версію додатку, так і мобільний застосунок, а також інші клієнтські сервіси. Додатковою перевагою є те, що компоненти системи можуть масштабуватися незалежно один від одного, що суттєво полегшує обслуговування та розвиток проєкту [4].

Окрім цього, використання статичного рендерингу (SSG) або серверного рендерингу (SSR) дозволяє значно підвищити продуктивність системи, зменшити час завантаження сторінок та покращити загальну взаємодію користувача з вебресурсом [5]. Завдяки попередньому рендерингу вмісту або його генерації на сервері лише за запитом, знижується навантаження на клієнтську частину та оптимізується споживання ресурсів. Важливою особливістю такого підходу є також підвищена безпека: CMS або інші backend-системи залишаються недоступними ззовні, оскільки взаємодія з ними відбувається виключно через заздалегідь налаштовані, захищені API. Це мінімізує ризики несанкціонованого доступу до адміністративної панелі або баз даних і створює додатковий захисний бар'єр у структурі додатку.

Висновок

Отже, в результаті дослідження було встановлено, що Headless-архітектура є найбільш ефективним і перспективним підходом для розробки сучасних WEB-додатків. Вона забезпечує гнучкість у розробці інтерфейсів, незалежне масштабування компонентів та підтримку мультиканальних рішень, що особливо важливо в умовах зростаючих вимог до продуктивності та користувацького досвіду. Використання API для взаємодії між CMS і клієнтською частиною дозволяє легко інтегрувати різні платформи та застосовувати передові технології рендерингу (SSR, SSG), що підвищує швидкість завантаження та безпеку системи. Таким чином, Headless-підхід є оптимальним вибором для створення універсальних, масштабованих та безпечних WEB-додатків нового покоління.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Headless CMS – нові можливості для розробки сайтів у 2025 році. Доступно за посиланням: https://the-complex.agency/blog/headless-cms-novi-mozhlivosti-dlya-rozrobki-sajtiv-u-2025-roci/
2. Understanding Full Stack Development Architecture: A Comprehensive Guide / P. Reaboi. Medium, 2021. URL: https://medium.com/@p.reaboi.frontend/understanding-full-stack-development-architecture-a-comprehensive-guide-548f8cba6d91
3. CMS – Вікіпедія HostIQ.ua. URL: https://hostiq.ua/wiki/ukr/cms/
4. What is a Headless CMS? DreamHost Blog. URL: https://www.dreamhost.com/blog/uk/headless-cms/
5. Форум DOU.ua: Headless CMS, обговорення та досвід використання. URL: https://dou.ua/forums/topic/41585/

Кононенко Назарій Валерійович – студент групи 2КН-21б, факультет інтелектуальних інформаційних технологій та автоматики, Вінницький національний технічний університет, Вінниця, e-mail: nazarella8@gmail.com.

Петришин Сергій Іванович – кандидат технічних наук, старший викладач кафедри комп'ютерних наук, Вінницький національний технічний університет, м. Вінниця, e-mail: petryshyn@vntu.edu.ua.

Сімончук Сергій Володимирович – асистент кафедри комп'ютерних наук, Вінницький національний технічний університет, Вінниця, e-mail: sergii.simonchuk@vntu.edu.ua.

Nazarii V. Kononenko – Faculty of intelligent information technologies and automation, Vinnytsia National Technical University, Vinnytsia, e-mail: nazarella8@gmail.com.

Petryshyn Serhii Ivanovych – PhD, Senior Lecturer of the Department of Computer Science, Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: petryshyn@vntu.edu.ua.

Simonchuk Serhii Volodymyrovych – Assistant Lecturer of the Department of Computer Science, Vinnytsia National Technical University, Vinnytsia, Ukraine, e-mail: sergii.simonchuk@vntu.edu.ua.