Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
https://doi.org/10.36930/40360307
Article received: 23.03.2026
Article accepted: 25.05.2026
Article published: 29.06.2026
Correspondence author: R. B. Tushnytskyy ruslan.b.tushnytskyy@lpnu.ua

UDC 004.[738.5+415+89]

О. Ю. Устич, Р. Б. Тушницький
Національний університет "Львівська політехніка", м. Львів, Україна

СИСТЕМА КЕРУВАННЯ ВМІСТОМ ВЕБСАЙТУ НА ОСНОВІ МІКРОСЕРВІСНОЇ ЙОГО АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

З'ясовано, що сучасний ринок веброзроблення стикається зі значним технологічним бар'єром для користувачів без навичок програмування, що обмежує їхні можливості у створенні придатних для роботи ресурсів без залучення дорогих спеціалістів. Оцінено вплив технологій штучного інтелекту, зокрема – великих мовних моделей, на процеси створення цифрового контенту та автоматичного генерування програмного коду для сучасних вебзастосунків. Виникає потреба розроблення інноваційної автономної системи керування вмістом вебсайтів, що інтегрує штучний інтелект у власну мікросервісну архітектуру застосунку для автоматизації рутинних процесів розроблення вебсайтів "з нуля". Встановлено, що наявні рішення на ринку (Wix, Webflow, Builder.io) пропонують компроміс між гнучкістю та простотою використання, обмежуючи користувачів шаблонними рамками або, навпаки, вимагаючи глибоких технічних знань. Охарактеризовано закономірності побудови систем керування вмістом вебсайтів на основі мікросервісної архітектури програмного комплексу з використанням платформи оркестрації.NET Aspire, що забезпечує високий рівень масштабованості всієї системи, незалежності компонентів та стійкості до відмов. Виявлено, що використання платформи Gemini дає змогу автоматизувати процес створення HTML- та CSS-коду з високою точністю генерування на основі запитів природною мовою завдяки оптимальному співвідношенню витрат токенів. Розроблено багаторівневу архітектуру програмного комплексу, яка містить чотири основні мікросервіси: FrontendReactVite для інтерфейсу, DataService для роботи з SQL Server, WebFrontEnd для рендерингу через шаблонізатор Razor Engine та AIService для семантичного аналізу. Досліджено ефективність застосування бази шаблонів запитів (prompt templates) та підходу до інженерії промптів "few-shot prompting" у спеціалізованому модулі PromptEngineer для вдосконалення взаємодії зі штучним інтелектом. Встановлено, що розроблена реляційна модель даних забезпечує гнучке управління різними типами контенту та їхніми динамічними властивостями без надлишковості коду. Перспективи подальших етапів дослідження полягають у розширенні спектра підтримуваних моделей штучного інтелекту, вдосконаленні алгоритмів генерування складних динамічних вебкомпонентів та інтеграції розширеної аналітики взаємодії користувачів із згенерованим контентом.

Ключові слова: вебтехнології; мікросервіси; генерування коду; штучний інтелект; Gemini;.NET Aspire; React; інженерія запитів.

Вступ / Introduction

На сучасному етапі розвитку інформаційних технологій наявність якісного вебсайту або іншого онлайн-представництва стала важливою умовою ефективної діяльності компаній різного масштабу та просування індивідуальних проєктів. Водночас, для представників малого й середнього бізнесу створення та підтримка таких рішень часто пов'язані зі значними фінансовими витратами на послуги професійних веброзробників. Це зумовлює потребу в інструментах, які спрощують процес розроблення вебресурсів і дають змогу створювати базові онлайн-рішення без залучення значних технічних та фінансових ресурсів.

Система керування вмістом CMS (англ. *Content Management System*) є спеціалізованим програмним забезпеченням [21], що дає змогу пересічним користувачам створювати, редагувати та публікувати цифровий контент без необхідності писати програмний код вручну, виконуючи роль проміжного ланцюга між користувачем і безпосереднім сервером.

Попри наявність таких компонентів, як сторінка адміністратора, редактор вмісту, база даних та менеджер шаблонів, традиційні CMS мають ряд істотних обмежень. Однією з основних проблем є дисбаланс між простотою використання та функціональними можливостями таких систем. З одного боку, прості CMS є зручними для освоєння користувачами без спеціальної технічної підготовки, однак часто обмежують процес створення вебресурсів фіксованими шаблонами, що може призводити до надлишкового або недостатньо оптимізованого коду. З іншого боку, гнучкіші CMS надають ширші можливості налаштування, проте потребують спеціальних знань у сфері веброзроблення, що створює істотний технологічний бар'єр для підприємців, маркетологів та інших нетехнічних користувачів.

Водночас, технології штучного інтелекту (ШІ) досягли такого рівня, на якому вони здатні глибоко аналі-

© Copyright 2026 by the author(s)

Устич Олександр Юрійович, бакалавр, кафедра програмного забезпечення.
Email: oleksandr.ustych.pz.2021@lpnu.ua; https://orcid.org/0009-0009-9429-6845
Тушницький Руслан Богданович, канд. техн. наук, доцент, кафедра програмного забезпечення.
Email: ruslan.b.tushnytskyy@lpnu.ua; https://orcid.org/0000-0002-8522-0293

Цитування за ДСТУ: Устич О. Ю., Тушницький Р. Б. Система керування вмістом вебсайту на основі мікросервісної його архітектури та технологій штучного інтелекту. Науковий вісник НЛТУ України. 2026, т. 36, № 3. С. 72–78.
Citation APA: Ustych, O. Yu., & Tushnytskyy, R. B. (2026). Content management system for websites based on microservice architecture and artificial intelligence technologies. *Scientific Bulletin of UNFU*, 36(3), 72–78. https://doi.org/10.36930/40360307

зувати контекст запитів, розуміти наміри користувача та генерувати семантично чистий та досконалий код (HTML, CSS). Проте, інтеграція ШІ в архітектуру сучасних СКВ досі здебільшого обмежується допоміжними функціями, а не фундаментальним генеруванням структури "з нуля".

Об'єкт дослідження – розроблення та функціонування системи керування вебконтентом.

Предмет дослідження – архітектурні методи, моделі даних та програмні засоби інтеграції технологій штучного інтелекту у мікросервісну архітектуру CMS, які дадуть змогу автоматизувати процеси генерування та розгортання вебзастосунків.

Мета роботи – розробити систему керування вмістом вебсайту на основі мікросервісної його архітектури з інтеграцією технологій штучного інтелекту, що мінімізує потребу в ручному написанні HTML/CSS та знизить поріг входження в середовище розроблення для нетехнічних фахівців.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- розробити гнучку та надійну мікросервісну архітектуру системи керування вмістом вебсайтів на базі програмного каркаса.NET Aspire [17], що дасть змогу підвищити її стійкість до відмов та спростити оркестрацію сервісів у хмарному середовищі;
- створити інтуїтивно зрозумілий та високопродуктивний користувацький інтерфейс для адміністративної панелі за допомогою бібліотеки React [9], що дасть змогу реалізувати концепцію Single Page Application та забезпечити швидку реакцію її інтерфейсу на дії користувача;
- забезпечити комплексну інтеграцію проєктованої системи керування вмістом вебсайтів з сервісами ШІ (на базі моделі Gemini) для автоматизованого генерування програмного коду на основі запитів природною мовою [1], що мінімізує потребу в ручному написанні HTML/CSS та знижує поріг входження в середовище розроблення для нетехнічних фахівців;
- забезпечити надійне зберігання даних з використанням SQL Server та впровадження Razor Engine [12] для швидкого генерування динамічних вебсторінок, що дасть змогу відтворювати фінальні вебресурси з мінімальними затримками.

Аналіз останніх досліджень та публікацій. Наразі проблема інтеграції генеративного штучного інтелекту (ШІ) у процеси розроблення вебзастосунків та створення систем керування вмістом (CMS) вебсайтів набуває значної актуальності. Інтеграція генеративного ШІ у веброзробку та системи керування вмістом створює проблеми з безпекою коду, точністю даних та високою вартістю інфраструктури. Інформаційні технології допомагають працювати швидше, але приносять нові складні виклики для розробників і бізнесу.

У роботі [14] автори проаналізували фреймворк інтелектуального генерування вебсайтів і довели, що системи веброзроблення на базі ШІ здатні автоматизувати процес написання коду та здійснити персоналізацію контенту, істотно скорочуючи тривалість розроблення порівняно з традиційними підходами.

У дослідженні [18] проаналізовано ефективність "AI Website Builder" та підтверджено, що використання моделей-трансформерів дає змогу ефективно перетворювати запити природною мовою у функціональні вебсайти з точністю мапінгу 83 %.

У роботі [11] запропоновано воркфлоу-орієнтовану архітектуру системи багатоплатформного управління контентом CLAW, у якій відокремлення базових мікросервісів від модуля взаємодії зі ШІ забезпечує високу масштабованість цієї системи та стабільну оркестрацію процесів.

У дослідженні [8] автори вказали на те, що інтеграція великих мовних моделей у життєвий цикл розроблення програмного забезпечення потребує управління ризиками, пов'язаними з безпекою, коректністю та надійністю згенерованого коду, а також створення спеціалізованих модулів його валідації.

У роботі [2] досліджено вплив інструментів GenAI на створення вебзастосунків користувачами без технічного бекграунду, що вказало на гостру потребу у впровадженні проміжних механізмів "перекладу" абстрактних вимог користувача на формалізовані технічні інструкції для подальшого генерування програмного коду.

У дослідженні [4] розглянуто метод генерування вебресурсів, який використовує інженерію запитів для примусового дотримання моделлю специфічного шаблону HTML/CSS коду, що дає можливість оновлювати тільки окремі частини документа.

Аналіз останніх досліджень та публікацій свідчить про те, що попри значні теоретичні напрацювання, наразі недостатньо дослідженими залишаються комплексні рішення, які поєднують автоматизоване генерування структури вебресурсів за допомогою великих мовних моделей із перевагами мікросервісної архітектури систем керування вмістом вебсайтів. Більшість наявних AI-орієнтованих інструментів створення вебресурсів обмежуються або шаблонними рішеннями, або вимагають високої технічної кваліфікації користувача. Саме тому існує потреба у розробленні нового підходу до побудови інтелектуальної системи керування вмістом вебсайтів, який дасть змогу автоматизувати процес формування структури сайту та здійснити перетворення абстрактного запиту користувача на фінальний програмний код, придатний для використання.

Матеріали та методи дослідження. Дослідження базується на використанні мікросервісної архітектури проєктованої системи керування вмістом вебсайтів під управлінням .NET Aspire для забезпечення масштабованості та надійності роботи її сервісів. Для побудови користувацького інтерфейсу адміністративної панелі використано бібліотеку React. Взаємодію з базою даних SQL Server реалізовано через Entity Framework Core з використанням патерну Repository, а рендеринг динамічних сторінок здійснюється за допомогою шаблонізатора Razor Engine. Ключовим методом інтеграції ШІ обрано платформу Gemini, ефективність якої підтверджено експериментально за критеріями якості коду та мінімальних витрат токенів. Для вдосконалення відповідей ШІ застосовано техніку "few-shot prompting" у межах інженерії запитів, а надійність міжсервісних комунікацій перевірена у спосіб їх інтеграційного тестування у середовищі MSTest.

Результати дослідження та їх обговорення / Research results and their discussion

Мікросервісна архітектура застосунку. Для реалізації проєктованої системи керування вмістом вебсайтів розроблено розподілену архітектуру (рис. 1), що складається з чотирьох взаємопов'язаних мікросервісів, кожен з яких ізольовано виконує власну бізнес-функцію:

1. FrontendReactVite – клієнтський сервіс адміністративної панелі, що створений за допомогою бібліотеки Re-

act та збирача Vite. Він забезпечує візуальну взаємодію користувача із системою, обробляє дії, пов'язані зі створенням вебсторінок, редагуванням їхнього вмісту та керуванням структурою вебсайту, а також надсилає асинхронні HTTP-запити до DataService та AIService за допомогою бібліотеки axios. Інтеграцію React-застосунку в загальний простір .NET Aspire реалізовано через конфігурацію проксі-сервера у файлі vite.config.ts, що дало змогу керувати ним як окремим мікросервісом.

2. DataService – сервіс доступу до даних, реалізований як Web API на ASP.NET Core. У межах цього сервісу впроваджено універсальний базовий контролер BaseController<T> та репозиторій Repository<T>, що спростило подальше розширення моделі даних і додавання нових сутностей.
3. WebFrontEnd – сервіс генерування клієнтських сайтів. Він отримує з DataService вихідний код, згенерований ШІ для конкретного типу документа, і динамічно підставляє у нього збережені значення властивостей за допомогою реалізованого алгоритму заміни (через пошук токенів вигляду {PropertyName}), після чого сторінка відтворюється у браузері користувача.
4. AIService – інтелектуальний мікросервіс, який забезпечує взаємодію проєктованої системи з API Gemini та відповідає за формування запитів до моделі штучного інтелекту, отримання згенерованого програмного коду й передавання результатів для подальшого оброблення.

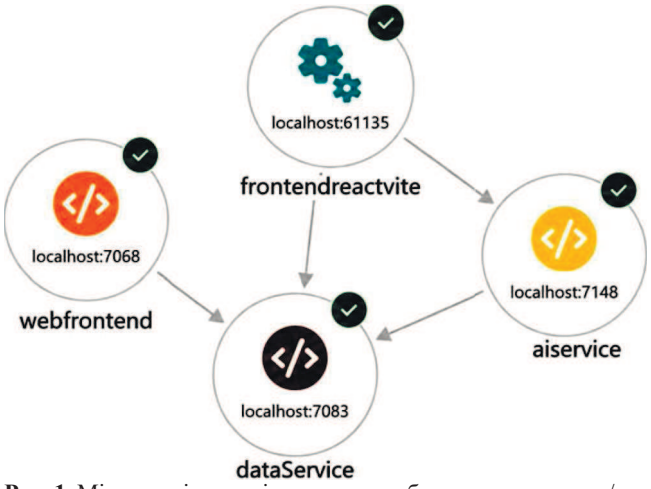

Рис. 1. Мікросервісна архітектура розробленого застосунку / Microservice Architecture of the Developed Application

Проєктування бази даних. Для гнучкого збереження згенерованого контенту та налаштувань спроєктовано реляційну модель БД (рис. 2), що містить такі основні сутності:

- сутність User (Користувач системи) зберігає дані користувача системи, який створює та адмініструє власні вебсайти для різних потреб;
- сутність Site (Вебсайт) містить налаштування конкретного вебсайту користувача, зокрема – його назву, домен, статус, дату останнього оновлення та параметри зображення для різних користувачів;
- сутність DocumentType (Тип документа) – визначає каркас сторінки (наприклад, "Блог", "Товар"). Ця сутність пов'язана з сутністю Code, яка зберігає єдиний HTML-шаблон для відповідного типу документа;
- сутність Property (Властивість) – описує, які поля (текст, зображення, метадані) містить певний DocumentType;
- сутність Content (Контент) – зберігає окремий екземпляр певного типу документа, наприклад конкретну статтю або конкретний товар;
- сутність ContentProperty є проміжною таблицею, що зберігає значення властивостей вебсайту, наприклад текст статті, і пов'язує сутність Content із відповідними сутностями Property;
- сутності Chat та Message призначені для збереження хронології запитів користувача до штучного інтелекту та відповідей моделі.

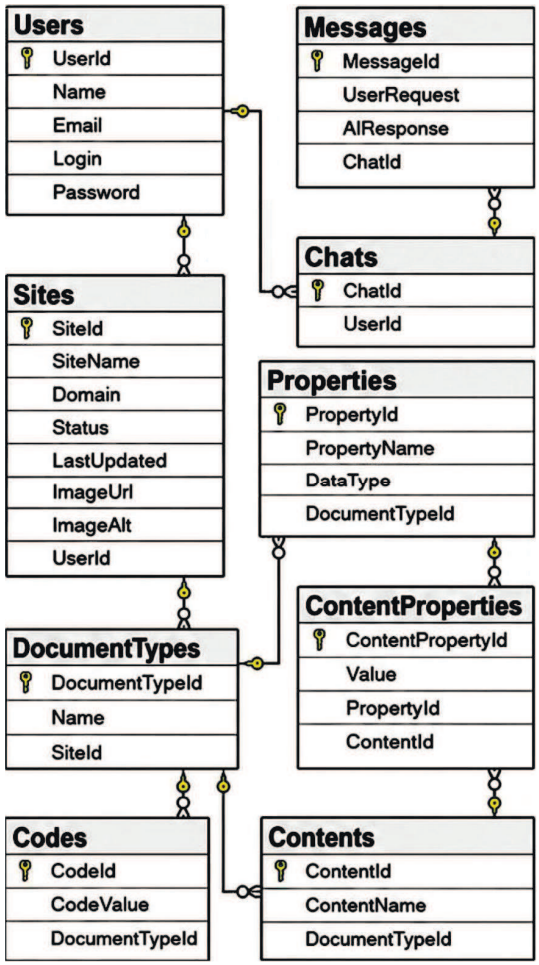

Рис. 2. Розроблена модель бази даних застосунку / Database model for the application has been developed

Така структура дає змогу користувачу створювати необмежену кількість вебсторінок на основі одного згенерованого ШІ-шаблону (DocumentType/Code), що гарантує високу продуктивність та відсутність дублювання коду в базі даних.

Порівняльний аналіз дослідження ключових функціональних та архітектурних параметрів сучасних CMS (Wix, Webflow, Builder.io) та розробленої авторами наведено в табл. 1.

Табл. 1. Порівняння функціональних та архітектурних параметрів сучасних CMS / Comparison of functional and architectural parameters of modern CMS platforms

Параметр	CMS з ШІ	Wix	Webflow	Builder.io
Генерування коду за допомогою ШІ	Вбудована	Вбудована	Вбудована	Вбудована
Складність для нетехнічних користувачів	Низька	Низька	Висока	Висока
Продуктивність	Висока	Середня	Висока	Висока
Масштабованість	Дуже висока	Обмежена	Висока	Дуже висока
Вартість володіння	Низька	Низька	Висока	Висока
Гнучкість налаштування	Середня	Середня	Висока	Середня
Швидкість розроблення	Дуже висока	Висока	Середня	Середня

У такий спосіб обґрунтовано наявність вільної ніші на ринку: існує потреба в системі, яка функціонуватиме

як міст між простотою Wix та професійною потужністю Webflow. Спроєктована система керування вмістом вебсайтів дає змогу зменшити залежність від фіксованих шаблонів без потреби вивчення синтаксису мов програмування, оскільки завдання формування складної структури вебресурсу передається інтегрованому модулю штучного інтелекту.

Алгоритми оброблення запитів ШІ. AIService – загальна назва хмарних інструментів, платформ та API, які надають розробникам доступ до готових моделей ШІ без необхідності створювати їх з нуля. Прикладами є генерування тексту, розпізнавання мовлення, комп'ютерний зір та автоматизація процесу оброблення великих обсягів документів.

Архітектура модуля AIService є багатокомпонентною і складаються з чотирьох основних блоків (рис. 3):

- блок RequestHandler – приймає вхідні текстові запити від користувача та виконує їх первинну валідацію;
- блок PromptEngineer – виконує основну логіку оброблення запиту. На першому етапі він надсилає до моделі штучного інтелекту короткий запит для класифікації наміру користувача, тобто визначає, чи потребує вхідний запит генерування HTML-коду, чи створення звичайного текстового вмісту. Після класифікації наміру користувача алгоритм звертається до розробленої бази шаблонів запитів, яка функціонує як бібліотека попередньо створених контекстуальних модифікаторів, зокрема – вказівок щодо семантики, SEO-параметрів і стилістики. На основі отриманих даних PromptEngineer формує розширений запит за стратегією few-shot prompting, налаштовуючи модель на очікуваний формат результату;
- блок AIConnector – забезпечує захищене HTTP-з'єднання з API Gemini та передавання сформованого запиту до моделі штучного інтелекту;
- блок CodeResponseProcessor – отримує відповідь моделі штучного інтелекту, ізолює блок програмного коду, вилучаючи зайві пояснення, перевіряє його цілісність і форматує результат для подальшого запису в базу даних.

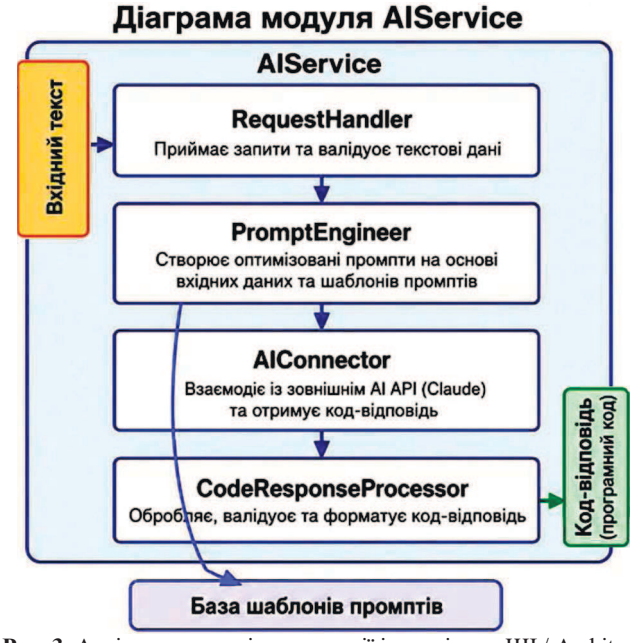

Рис. 3. Архітектура сервісу взаємодії із сервісами ШІ / Architecture of the AI services interaction service

Стратегія інтеграції штучного інтелекту в проєктовану систему керування вмістом вебсайтів формувалася на основі порівняльного дослідження трьох провідних моделей: ChatGPT, Gemini та Claude. Тестування моделей проводилось за критеріями якості генерованого коду та економічності (кількість використаних токенів). Експеримент базувався на трьох послідовних запитах: визначення контексту проєктованої системи керування вмістом вебсайтів, підготовчому запиті з деталізацією вимог та основному завданні на генерування програмного коду. Результати порівняння показали, що модель Gemini в межах проведеного експерименту виявилася найефективнішою: на опрацювання запиту витрачалося 75-85 токенів, на створення відповіді – 750-850 токенів, при цьому згенерований код мав найвищі показники семантичної коректності та відповідності поставленому завданню.

Оцінювання економічної ефективності проєкту. У межах дослідження було проведено оцінювання економічної доцільності розгортання розробленої CMS для підприємця, що планує надавати послуги за моделлю SaaS. Розрахунок точки беззбитковості (відношення постійних витрат до вартості підписки) показав, що для покриття всіх операційних витрат достатньо мати тільки 14 активних підписників. Рентабельність проєкту (відношення чистого прибутку до прибутку до оподаткування) становить надзвичайно високий показник – 85,81 %. Це свідчить про економічну доцільність розгортання системи керування вмістом вебсайтів та її комерційний потенціал навіть за умови мінімального охоплення цільової аудиторії.

Розроблена система керування вмістом вебсайтів успішно долає низку обмежень традиційних CMS і монолітних архітектур вебзастосунків. Вибір мікросервісного підходу до побудови архітектури системи, хоч і ускладнив етап її початкового налаштування, однак забезпечив високу ізольованість функціональних процесів. Наприклад, якщо сервіс генерування клієнтських вебсторінок (WebFrontEnd) зазнає пікового навантаження від відвідувачів створених сайтів, це ніяк не вплине на продуктивність адміністративної панелі (FrontendReactVite) або стабільність генерування програмного коду у модулі AIService.

Інтеграція ШІ-моделі Gemini через спеціалізований модуль PromptEngineer довела гіпотезу про можливість генерування чистого та семантично правильного коду. Використання попередньої класифікації запиту та бази шаблонів (contextual modifiers) кардинально відрізняє розроблений підхід від простого "прокидання" тексту користувача до API. Це дало змогу перетворити проєктовану систему керування вмістом вебсайтів із простого інтерфейсу взаємодії з чат-моделлю на прикладний інструмент автоматизованого створення вебресурсів, який враховує стандарти написання HTML, принципи адаптивного дизайну та вимоги SEO-оптимізації. Водночас, мінімізація кількості використаних токенів знижує витрати провайдера SaaS-платформи.

Проте, потрібно враховувати потенційні недоліки. Зі збільшенням кількості мікросервісів та зростанням навантаження можуть виникнути труднощі у забезпеченні узгодженості даних та збільшитися операційні витрати на підтримання серверної інфраструктури, що вимагатиме переходу до повноцінних хмарних провайдерів та кластерних рішень (Kubernetes).

Запропонована модель бази даних із динамічними властивостями є значно гнучкішою за жорстку фіксацію полів у реляційних таблицях, проте, за великих обсягів даних може вимагати додаткового кешування для пришвидшення роботи рендерера динамічних вебсторінок.

Проведені розрахунки підтверджують комерційну привабливість розгортання розробленої системи керування вмістом вебсайтів за моделлю SaaS. Завдяки автоматизації значної частини процесів створення вебресурсів із використанням штучного інтелекту витрати на підтримку платформи залишаються порівняно низькими. Рентабельність розробленої системи на рівні 85,81 % свідчить про можливість швидкого повернення початкових інвестицій і подальшого фінансування заходів із масштабування проєкту.

Обговорення результатів дослідження. Результати аналізу наявних на ринку систем керування вмістом вебсайтів показали, що більшість популярних рішень функціонують у парадигмі компромісу між доступністю та технічною гнучкістю.

У дослідженні платформи Wix [13] автори проаналізували можливості використання візуального drag-and-drop редактора та інтегрованих засобів штучного інтелекту для автоматизації створення вебресурсів. Встановлено, що платформа Wix забезпечує низький поріг входження в середовище розроблення для нетехнічних користувачів, підтримує автоматизоване налаштування структури вебсторінок і генерування контенту. Водночас, більшість механізмів штучного інтелекту цієї платформи функціонує в межах шаблонно-орієнтованої архітектури створення вебсайтів, що обмежує гнучкість налаштування структури вебресурсу, його масштабованість та можливість формування унікального програмного коду для складніших вебзастосунків.

У дослідженні платформи Webflow [22] автори розглянули підходи до створення професійних вебінтерфейсів із використанням візуального середовища проєктування та генерування семантично коректного HTML / CSS-коду. Встановлено, що платформа Webflow забезпечує високий рівень контролю над дизайном, продуктивністю та SEO-оптимізацією вебресурсів. Водночас, ефективне використання цієї платформи потребує від користувача глибших знань принципів веброзроблення, зокрема – HTML та CSS, що створює технологічний бар'єр для нетехнічних користувачів і підвищує загальну вартість володіння відповідним інструментом створення вебресурсів.

У дослідженні платформи Builder.io [3] автори проаналізували застосування концепції headless CMS для побудови масштабованих вебсистем корпоративного рівня. Результати аналізу показали, що платформа забезпечує інтеграцію візуально створених компонентів у наявні вебзастосунки через API, підтримує високу масштабованість та гнучкість архітектури вебсистеми. Водночас, встановлено, що платформа Builder.io орієнтована переважно на enterprise-сегмент і професійні команди розробників, потребує складного налаштування та додаткової серверної інфраструктури, а також не забезпечує повноцінного автономного створення вебресурсів користувачами без технічної підготовки.

Загальні закономірності розвитку архітектури інформаційних систем розглянуто у роботі [7]. Автори показують, що еволюція архітектури інформаційних систем пов'язана з переходом від жорстко зв'язаних структур до більш адаптивних і розподілених моделей. Це узгоджується з архітектурним рішенням, запропонованим у цій роботі, де функції зберігання даних, адміністрування, рендерингу вебсторінок і взаємодії зі штучним інтелектом винесено в окремі мікросервіси. Такий підхід підвищує гнучкість системи та створює передумови для її подальшого масштабування.

Окремий напрям сучасних досліджень пов'язаний із використанням штучного інтелекту у веброзробленні. У роботі [15] показано, що ШІ поступово змінює підходи до створення вебтехнологій, впливаючи на персоналізацію, інтерактивність, доступність і автоматизацію процесів розроблення. Це підтверджує актуальність інтеграції генеративного ШІ безпосередньо у функціональне ядро систем керування вебконтентом, а не тільки використання його як допоміжного інструмента для створення текстів або окремих елементів дизайну.

У праці [24] обґрунтовано використання бізнес-аналітики для трансформації бізнес-моделей малих і середніх підприємств за умов обмеженого часу. Хоча це дослідження не стосується безпосередньо CMS, його результати є важливими для обґрунтування практичної цінності запропонованої системи. Розроблена CMS може бути корисною саме для малого й середнього бізнесу, оскільки дає змогу зменшити витрати на створення вебресурсів, швидше запускати онлайн-представництва та адаптувати цифровий контент до потреб користувачів без постійного залучення професійних веброзробників.

У систематичному огляді [5] проаналізовано можливості, виклики та перспективи використання штучного інтелекту в освітньому середовищі. Незважаючи на освітній фокус цієї роботи, її висновки щодо необхідності відповідального використання ШІ, контролю якості результатів і врахування ризиків автоматизованого генерування контенту є релевантними для цього дослідження. У запропонованій CMS відповідь моделі Gemini не використовується безпосередньо, а проходить додаткове оброблення в межах AIService, що знижує ризики отримання некоректного або неповного HTML/CSS-коду.

Подібні аспекти розглянуто у роботі [10], де проаналізовано можливості й обмеження великих мовних моделей, зокрема ChatGPT. Автори наголошують, що LLM можуть істотно підтримувати процеси створення текстового й навчального контенту, але водночас потребують критичного оцінювання результатів, перевірки достовірності та контролю якості згенерованих відповідей. Це безпосередньо узгоджується з архітектурою розробленої системи, у якій передбачено окремі етапи формування запиту, отримання відповіді від моделі та подальшого оброблення згенерованого коду.

З погляду архітектури програмної системи, важливим є дослідження [19], у якому розглянуто проєктування та реалізацію онлайн-платформи на основі мікросервісної архітектури. Автори показують, що поділ системи на окремі сервіси сприяє кращій масштабованості, спрощує супровід програмного комплексу та дає змогу незалежно розвивати деякі функціональні компоненти. Це підтверджує доцільність реалізації розробленої CMS у вигляді окремих сервісів FrontendReactVite, DataService, WebFrontEnd та AIService.

Робота [20] стосується використання підходів code generation у контексті автоматизованого оцінювання відповідей природною мовою. Хоча предметна область цієї праці відрізняється від тематики CMS, вона демонструє ширше застосування автоматичного генерування коду для перетворення текстових формулювань у формалізовані програмні конструкції. Це опосередковано підтверджує перспективність підходу, використаного у цій роботі, де природномовний запит користувача перетворюється на HTML/CSS-код вебсторінки.

Важливим для цього дослідження є також напрям інженерії запитів. У роботі [6] prompt engineering розглянуто як нову важливу навичку роботи з генеративними моделями штучного інтелекту. Автори наголошують, що якість результатів, отриманих від LLM, значно залежить від структури запиту, контексту та точності формулювання очікуваного результату. Це підтверджує доцільність використання у розробленій системі бази шаблонів запитів і стратегії few-shot prompting, оскільки такі засоби дають змогу зменшити неоднозначність користувацьких вимог і сформувати більш передбачуваний результат у вигляді HTML/CSS-коду.

У праці [23] обґрунтовано розвиток експертного навчання в межах освітньої спільноти з кібербезпеки. Попри те, що ця робота не стосується безпосередньо систем керування вмістом вебсайтів, вона підкреслює важливість підготовки користувачів до роботи зі складними цифровими інструментами. Для запропонованої CMS це має опосередковане значення, оскільки система орієнтована на зменшення потреби у спеціальній технічній підготовці та надання користувачам можливості створювати вебресурси через природномовну взаємодію з модулем штучного інтелекту.

Обговорення результатів дослідження підтверджують доцільність виконання саме нашої роботи, оскільки сучасні інструменти створення вебресурсів або орієнтовані на просте шаблонне формування сайтів із обмеженими можливостями гнучкого налаштування, або потребують від користувача спеціальних знань у сфері веброзроблення. На відміну від таких рішень, у межах цього дослідження запропоновано систему керування вмістом вебсайтів, у якій поєднано мікросервісну архітектуру, динамічну модель даних і модуль інтелектуального генерування програмного коду на основі запитів природною мовою. Це дає змогу автоматизувати процес формування структури вебресурсу без жорсткої прив'язки до фіксованих шаблонів і без необхідності ручного написання HTML/CSS-коду нетехнічними користувачами.

Отже, внаслідок виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – удосконалено метод побудови інтелектуальної системи керування вмістом вебсайтів, який, на відміну від наявних методів, поєднує мікросервісну архітектуру, динамічну модель зберігання властивостей вебконтенту та модуль генерування HTML/CSS-коду на основі запитів природною мовою із використанням бази шаблонів запитів і стратегії few-shot prompting, що дає змогу автоматизувати процес створення структури вебресурсу, зменшити залежність користувача від фіксованих шаблонів і мінімізувати потребу в ручному написанні програмного коду.

Практична значущість результатів дослідження – можна застосувати для проєктування та впровадження інтелектуальних SaaS-рішень автоматизованого створення вебресурсів, зокрема – для малого й середнього бізнесу, освітніх проєктів, персональних сайтів та інших онлайн-представництв, де важливими є зменшення часових і фінансових витрат на веброзроблення, спрощення процесу керування вмістом вебсайтів і забезпечення можливості створення вебресурсів користувачами без спеціальної технічної підготовки.

Висновки / Conclusions

Розроблено систему керування вмістом вебсайту на основі мікросервісної його архітектури з інтеграцією технологій штучного інтелекту, яка мінімізує потребу в ручному написанні HTML/CSS та знижує поріг входження в середовище розроблення для нетехнічних фахівців. За результатами проведеного дослідження можна зробити такі основні висновки.

1. Систематизовано підходи до побудови відмовостійких вебсистем на базі фреймворку .NET Aspire та бібліотеки React, що забезпечило створення гнучкої інфраструктури з чітким розділенням бізнес-логіки між сервісами зберігання даних (DataService), генерування клієнтських вебсторінок (WebFrontEnd) та інтелектуального оброблення запитів (AIService).
2. Розроблено та імплементовано спеціалізований компонент PromptEngine, який на основі розширеної бази контекстуальних шаблонів запитів, моделі Gemini та стратегії "few-shot prompting" автоматизує процес генерування семантично коректного HTML/CSS коду та зменшує витрати токенів під час взаємодії з моделлю штучного інтелекту.
3. Розроблено реляційну модель бази даних, у якій динамічне зв'язування типів контенту зі згенерованими HTML-шаблонами вебсторінок забезпечує повторне використання програмного коду, зменшує його надлишковість і спрощує оброблення інформації про структуру вебресурсу.
4. Застосовано шаблонізатор Razor Engine для серверного генерування динамічних вебсторінок, що забезпечило відтворення результатів роботи модуля штучного інтелекту у браузері безпосереднього користувача.
5. Проведено оцінювання економічної ефективності розгортання розробленої системи керування вмістом вебсайтів за моделлю SaaS, за результатами якого встановлено, що точка беззбитковості досягається за 14 активних підписок на місяць, а прогнозована рентабельність проєкту становить 85,81 %.

Перспективи подальших етапів дослідження полягають у розширенні спектра підтримуваних ШІ-моделей (мультимодельна інтеграція), впровадженні функцій автоматичного A/B тестування згенерованого вмісту вебсторінок та розробленні алгоритмів для генерування складної бізнес-логіки мовою JavaScript для динамічних компонентів вебсайтів.

References

1. Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., & Firat, O. (2023). Google Gemini Team. Gemini: A family of highly capable multimodal models. *arXiv*. https://doi.org/10.48550/arXiv.2312.11805
2. Buendía-García, F., & Piris-Ruano, J. (2025). Using generative AI to support UX design students in web development courses. *Applied Sciences, 15*(13), article ID 7389. https://doi.org/10.3390/app15137389
3. Builder.io. (2026). *Visual development platform*. URL: https://builder.io
4. Calò, T., & De Russis, L. (2023). Leveraging large language models for end-user website generation. In L. D. Spano, A. Schmidt, C. Santoro, & S. Stumpf (Eds.), *End-user development: IS-EUD 2023* (Lecture Notes in Computer Science, 13917, 52–61). Springer. https://doi.org/10.1007/978-3-031-34433-6_4
5. Chiu, T. K. F., Xia, Q., Zhou, X., Chai, C. S., & Cheng, M. (2023). Systematic literature review on opportunities, challenges, and future research recommendations of artificial intelligence in education. *Computers and Education: Artificial Intelligence, 4*, article ID 100118. https://doi.org/10.1016/j.caeai.2022.100118

6. Federiakin, D., Molerov, D., Zlatkin-Troitschanskaia, O., & Maur, A. (2024). Prompt engineering as a new 21st century skill. *Frontiers in Education, 9*, article ID 1366434. https://doi.org/10.3389/feduc.2024.1366434
7. Haki, K., Beese, J., Aier, S., & Winter, R. (2020). The evolution of information systems architecture: An agent-based simulation model. *MIS Quarterly, 44*(1), 155–184. https://doi.org/10.25300/MISQ/2020/14494
8. Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology, 33*(8), article number 220, 1–79. https://doi.org/10.1145/3695988
9. Jonathan, R., & Suprihadi. (2023). Development of front-end web applications utilizing single page application framework and React.js library. *International Journal Software Engineering and Computer Science* (*IJSECS*), *3*(3), 529–536. https://doi.org/10.35870/ijsecs.v3i3.1943
10. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences, 103*, article ID 102274. https://doi.org/10.1016/j.lindif.2023.102274
11. Le, M. T., & Nguyen, T. T. (2026). A workflow-oriented architecture integrating large language models for automated multi-platform content management. *Ingénierie des Systèmes dInformation, 31*(1), 157–166. https://doi.org/10.18280/isi.310115
12. Lydford, S. (2011). Working with Razor and ASP.NET Web Pages. In: Building ASP.NET Web Pages with Microsoft WebMatrix. Apress. https://doi.org/10.1007/978-1-4302-4021-1_4
13. Madiah, M., Ng, K. X., Tan, Y. W., Tan, Z. H., Chong, Z. T., & Chan, J. X. (2024). Wix for web development and the application of the waterfall model and project based learning for project completion: A case study. *Journal of Informatics and Web Engineering*, *3*(2), 212–228. https://doi.org/10.33093/jiwe.2024.3.2.16
14. Maryam, S., Tahir, M., Paracha, M. F. K., & Paracha, W. T. (2026). *Intelligent website generation framework using artificial intelligence and data mining. Spectrum of Engineering Sciences, 3*(3), 1428–1438. https://doi.org/10.5281/zenodo.19347260
15. Mester, U., & Toth, A. (2025). The evolution of web development: The role of AI in shaping future web technologies. *Discover Artificial Intelligence, 5*, article ID 40. https://doi.org/10.1007/s44163-025-00567-4
16. Prompt Engineering Guide. (2026). *Few-shot prompting*. URL: https://www.promptingguide.ai/techniques/fewshot
17. Rael, D. (2025). Getting started with.NET Aspire: Build cloud-native and distributed applications with ease. Apress. https://doi.org/10.1007/979-8-8688-1521-8
18. Reddy, S. V., Bodhke, H., Patil, S., Gorad, R., Chavan, A., & Mall, A. K. (2025). *AI website builder*. International Journal of Creative Research Thoughts (IJCRT), *13*(10), 83–86. URL: https://www.ijcrt.org/papers/IJCRTBH02018.pdf
19. Ren, X., Wang, H., & Cai, T. (2023). Design and implementation of a microservices-based online learning platform. In *Proceedings of the 2023 2nd International Conference on Educational Innovation and Multimedia Technology* (*EIMT 2023*) (pp. 455–460). Atlantis Press. https://doi.org/10.2991/978-94-6463-192-0_60
20. Smith, D. H., IV, & Zilles, C. (2024). Code generation based grading: Evaluating an auto-grading mechanism for "explain-in-plain-in-English" questions. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (pp. 171–177). Association for Computing Machinery. https://doi.org/10.1145/3649217.3653582
21. Strekalova, Y. A., & Bouakkaz, M. (2022). Content management system (CMS). In *Encyclopedia of Big Data*, 208–211. Springer. https://doi.org/10.1007/978-3-319-32010-6_43
22. Vasylenko, V. M., Karpenko, M. I., Skybinskyi, A. S., & Huyda, O. H. (2024). Analysis of advantages and limitations of no-code platforms using Webflow.com as an example. *Scientific Notes of Taurida National V. I. Vernadsky University. Series: Technical Sciences*, *2*, 65–69. https://doi.org/10.32782/2663-5941/2024.2/09
23. Yonemura, K., Nakata, R., Kawai, H., Hashimoto, M., Otsuka, Y., Sato, H., Fujii, T., & Tokuyama, M. (2023). *Motivation in Teaching Expert Development Project by KOSEN Security Educational Community.* 2023 IEEE Global Engineering Education Conference (EDUCON), 1–9. https://doi.org/10.1109/EDUCON54358.2023.10125167
24. Zamani, E. D., Griva, A., & Conboy, K. (2022). Using business analytics for SME business model transformation under pandemic time pressure. *Information Systems Frontiers, 24*(4), 1145–1166. https://doi.org/10.1007/s10796-022-10255-8

O. Yu. Ustych, R. B. Tushnytskyy

Lviv Polytechnic National University, Lviv, Ukraine

CONTENT MANAGEMENT SYSTEM FOR WEBSITES BASED ON MICROSERVICE ARCHITECTURE AND ARTIFICIAL INTELLIGENCE TECHNOLOGIES

The contemporary web development market presents a significant technological barrier for non-programmer users, restricting their capacity to build professional web resources without deploying substantial financial capital for external software experts. This study evaluates the impact of artificial intelligence technologies, specifically large language models, on dynamic digital content creation and automated code generation workflows for modern web applications. To resolve this operational issue, there is a clear need to design an innovative autonomous website content management system that integrates artificial intelligence directly into its own application microservice architecture to automate routine website development processes from scratch. Current market alternatives (such as Wix, Webflow, and Builder.io) enforce a rigid trade-off between configuration flexibility and simplicity of use, either confining users within template frameworks or demanding advanced technical proficiency. This paper characterizes the structural regularities of building website content management systems based on a software suite microservice architecture orchestrated via the .NET Aspire platform, which ensures a high level of system-wide scalability, component independence, and fault tolerance. The experimental evaluation demonstrates that leveraging the Google Gemini platform successfully automates HTML and CSS code compilation with high generation fidelity directly from natural language queries, optimizing token-to-output consumption ratios. The investigation delivers a multi-tier software suite architecture isolating core business logic across four standalone microservices: FrontendReactVite for administrative state management, DataService for SQL Server persistence, WebFrontEnd for high-throughput rendering via the Razor Engine, and AIService for semantic processing. Furthermore, this research investigates the efficiency of deploying a dedicated prompt template database and a few-shot learning approach within a specialized PromptEngineer module designed to improve upstream human-AI interaction. The underlying relational database model enables flexible management of various content types and their dynamic properties without inducing data schema redundancy. Future research paths will focus on expanding the scope of supported foundation models, improving core algorithms for generating complex dynamic web components, and integrating advanced analytics tracking user interactions with the generated web content.

Keywords: web technologies; microservices; code generation; artificial intelligence; Gemini;.NET Aspire; React; prompt engineering.