

ПВНЗ «ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»

КАФЕДРА Комп'ютерних наук та програмної інженерії

СПЕЦІАЛЬНІСТЬ 121 Інженерія програмного забезпечення
(шифр та назва)

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА Інженерія програмного забезпечення
(назва)

**ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА
(ПРОЄКТ)**

на тему:

Розробка системи управління контентом (CMS) для малого бізнесу
(назва затвердженої теми)

На прикладі компанії з озелененням територій

(на прикладі _____)

(підпис студента)

Яківець Вадим Олександрович
(прізвище, ім'я по-батькові студента)

Науковий
керівник _____
(підпис наукового керівника)

Саморай О.К
(прізвище, ініціали, наукове звання і ступінь)

Допущено до захисту “ ____ ” _____ 2025 р.

Завідувач кафедри _____
(підпис)

(прізвище, ініціали, наукове звання і ступінь)

Київ - 2025

ПВНЗ «ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»

КАФЕДРА Комп'ютерних наук та програмної інженерії

СПЕЦІАЛЬНІСТЬ Інженерія програмного забезпечення

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА Інженерія програмного забезпечення

ЗАВДАННЯ

на випускню кваліфікаційну роботу (проєкту) студентів (ці):

Яківець Вадим Олександрович
(прізвище, ім'я, по-батькові студента (ки))

1. Тема роботи Розробка системи управління контентом (CMS) для малого бізнесу

2. Затверджена наказом по університету від “__” _____ 20__ р. №__

3. Цільова установка на вихідні дані до роботи

Провести огляд популярних CMS, обрати потрібні технології, створити власну просту CMS, яка дозволяє додавати, редагувати та видаляти контент. Продемонструвати роботу системи на прикладі невеликого бізнесу.

4. Перелік графічного матеріалу (з точним визначенням обов'язкових креслень) Схема структури сайту, приклад інтерфейсу, схема бази даних, діаграма роботи системи.

5. Консультанти по роботі, із зазначенням розділів, що їх стосуються:

Розділ	Науковий керівник	Підпис, дата	
		Завдання видав	Завдання отримав
Аналіз потреб та обґрунтування вибору технології	<u>Саморай О.К</u>	<u>15.01.2025</u>	<u>16.01.2025</u>
Проектування системи управління контентом	<u>Саморай О.К</u>	<u>28.02.2025</u>	<u>28.02.2025</u>

Реалізація тестування проекту	та <u>Саморай О.К</u>	<u>24.03.2025</u>	<u>24.03.2025</u>
-------------------------------------	--------------------------	-------------------	-------------------

6. Зміст випускної кваліфікаційної роботи (перелік питань)

1. Обґрунтування актуальності теми та визначення мети дослідження.
2. Аналіз потреб малого бізнесу у сфері цифрової присутності та управління контентом.
3. Огляд та критичний аналіз існуючих CMS-рішень: WordPress, Joomla, Drupal, Wix, OpenCart, Shopify, MODX.
4. Порівняння функціональності, гнучкості, безпеки та зручності популярних платформ.
5. Вибір та обґрунтування сучасних технологій для створення власної CMS (Python, Django, HTML/CSS, JS).
6. Формування функціональних та нефункціональних вимог до системи.
7. Проектування архітектури CMS з використанням UML-діаграм (класів, компонентів, діяльності, послідовностей).
8. Реалізація ключових модулів: авторизація, управління контентом, меню, медіа, SEO, мультимовність, зворотний зв'язок.
9. Проведення багаторівневого тестування CMS: функціонального, UX/UI, навантажувального.
10. Аналіз результатів тестування, виявлення недоліків, формування пропозицій щодо вдосконалення.
11. Загальні висновки та рекомендації щодо подальшого впровадження та масштабування системи.

7. Календарний план виконання випускної кваліфікаційної роботи

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	
		За планом	Фактично
1	Обрання теми випускної кваліфікаційної роботи (ВКР)	<u>Жовтень 2024</u>	<u>Жовтень 2024</u>
2	Опрацювання літератури та інших джерел для написання ВКР	<u>01.01.2025</u>	<u>01.01.2025-16.01.2025</u>
3	Виконання 1 розділу ВКР	<u>17.01.2025</u>	<u>19.01.2025-</u>

			<u>25.02.2025</u>
4	Виконання 2 розділу ВКР	<u>30.02.2025</u>	<u>29.02.2025-</u> <u>22.03.2025</u>
5	Виконання 3 розділу ВКР, висновки	<u>24.03.2025</u>	<u>24.03.2025-</u> <u>13.04.2025</u>
6	Передзахист ВКР	<u>21.04.2025-</u> <u>02.05.2025</u>	<u>21.04.2025-</u> <u>02.05.2025</u>
7	Проведення перевірки на плагіат ВКР	<u>02.06.2025</u>	<u>02.06.2025</u>
8	Відгук керівника, рецензія	<u>12.06.2025</u>	<u>12.06.2025</u>

8. Дата видачі завдання “__” _____ 202_ р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

АНОТАЦІЯ

Дипломна робота присвячена розробці системи управління контентом (CMS), орієнтованої на потреби малого бізнесу. В умовах цифровізації економіки та зростання вимог до ефективної онлайн-присутності підприємств, виникає потреба у створенні доступних та функціональних рішень для управління веб-контентом без залучення спеціалістів із програмування.

Метою роботи є створення індивідуального програмного продукту, що дозволяє представникам малого бізнесу самостійно адмініструвати сайт, оновлювати інформацію, керувати структурою ресурсу та контентом. У процесі дослідження проведено аналіз сучасних CMS-систем, визначено їхні переваги й недоліки, сформульовано вимоги до функціональності та архітектури власної розробки.

На основі результатів аналізу спроектовано та реалізовано CMS, що включає зручний інтерфейс адміністратора, модулі управління сторінками, користувачами, медіаконтентом та SEO-налаштуваннями. Особливу увагу приділено безпеці, масштабованості системи та можливості подальшого розширення функціоналу.

Розроблений програмний продукт впроваджено та протестовано на прикладі компанії, що спеціалізується на озелененні територій, що дозволило підтвердити практичну цінність і ефективність запропонованого рішення.

Ключові слова: CMS, система управління контентом, малий бізнес, веб-розробка, програмне забезпечення, інформаційні технології.

ABSTRACT

The bachelor's thesis is devoted to the development of a content management system (CMS) focused on the needs of small businesses. In the context of the digitalization of the economy and the growing demand for an effective online presence, there is a need to create affordable and functional tools for managing web content without the involvement of programming specialists.

The purpose of this study is to create a custom software product that allows small business representatives to independently manage a website, update information, and control the structure and content of the resource. During the research, existing CMS solutions were analyzed, their strengths and weaknesses were identified, and requirements for the functionality and architecture of the new system were formulated.

Based on the analysis results, a CMS was designed and implemented, including an intuitive admin interface and modules for managing pages, users, media content, and SEO settings. Special attention was paid to system security, scalability, and the possibility of further functional expansion.

The developed product was implemented and tested on a landscaping company case study, which confirmed the practical value and effectiveness of the proposed solution.

Keywords: CMS, content management system, small business, web development, software, information technology.

ЗМІСТ

<u>ВСТУП</u>	6
<u>РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ</u>	9
<u>1.1. Аналіз потреб малого бізнесу в управлінні контентом</u>	9
<u>1.2. Аналіз існуючих рішень</u>	13
<u>1.2.1 WordPress</u>	13
<u>1.2.2. Joomla</u>	15
<u>1.2.3. Drupal</u>	17
<u>1.2.4. Wix</u>	18
<u>1.2.5. OpenCart</u>	20
<u>1.2.6. Shopify</u>	21
<u>1.2.7. MODX</u>	23
<u>1.3. Обґрунтування вибору технологій для розробки CMS</u>	25
<u>Висновки до розділу 1</u>	28
<u>РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ</u>	30
<u>2.1. Функціональні вимоги до CMS</u>	30
<u>2.2. Архітектура системи</u>	33
<u>2.3. Інтерфейс користувача</u>	43
<u>Висновки до розділу 2</u>	47
<u>РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТУ</u>	50
<u>3.1. Розробка основних модулів CMS</u>	50
<u>3.1.1. Модуль авторизації та аутентифікації користувачів</u>	51
<u>3.1.2. Модуль управління сторінками</u>	52
<u>3.1.3. Модуль управління меню та навігацією</u>	53
<u>3.1.4. Модуль мультимедіа</u>	53
<u>3.1.5. Модуль коментарів та зворотного зв'язку</u>	54
<u>3.1.6. Модуль мультимовності</u>	54
<u>3.1.7. Модуль пошукової оптимізації (SEO)</u>	54
<u>3.1.8. Модуль аналітики та статистики</u>	55
<u>3.1.9. Модуль безпеки</u>	55
<u>3.2. Тестування системи</u>	56
<u>3.2.1. Покрокове виконання проєкту</u>	56

	9
<u>3.2.2. Тестування запуску системи та перевірка головної сторінки</u>	70
<u>3.2.3. Перевірка навігації та переходів між сторінками</u>	72
<u>3.2.4. Тестування форми зворотного зв'язку</u>	73
<u>3.2.5. Тестування багатомовності системи</u>	74
<u>3.2.6. Тестування відображення динамічного контенту</u>	75
<u>3.2.7. Тестування навантаження та стабільності</u>	77
<u>3.3. Проблеми та перспективи вдосконалення CMS</u>	79
<u>3.3.1. Проблеми, що виникають у процесі розробки та використання CMS</u>	80
<u>3.3.2. Перспективи вдосконалення CMS</u>	81
<u>Висновки до розділу 3</u>	84
<u>ВИСНОВКИ</u>	87
<u>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ</u>	90
<u>Додаток А Лістинг програми</u>	95

ВСТУП

В умовах стрімкого розвитку цифрових технологій та зростання конкуренції на ринку, малий бізнес потребує ефективних інструментів для управління онлайн-контентом, що сприятимуть підвищенню видимості, залученню клієнтів та оптимізації внутрішніх процесів. Одним із таких інструментів є системи управління контентом (CMS), які забезпечують можливість створення та адміністрування вебресурсів без залучення спеціалізованих технічних фахівців.

Значна кількість існуючих CMS платформ (наприклад, WordPress, Joomla, Wix) надають широкий функціонал, однак не завжди адаптовані до потреб малого бізнесу, оскільки можуть бути або занадто складними, або надто обмеженими. У цьому контексті розробка нової CMS, орієнтованої на потреби малого бізнесу, із врахуванням таких критеріїв, як простота використання, безпека, гнучкість та адаптивність, є актуальною задачею. Особливого значення набуває інтеграція можливостей SEO-оптимізації, підтримка багатомовності та засоби зворотного зв'язку, які дозволяють бізнесу ефективно комунікувати з клієнтами та розширювати свою аудиторію.

Розробка системи управління контентом у даній роботі враховує сучасні тенденції ринку, досвід наявних рішень та вимоги до простоти і безпеки. Зв'язок дослідження з планами кафедри обумовлений її участю у проєктах із впровадження ІТ-рішень для малого бізнесу. Роль автора полягала у формулюванні технічних вимог до системи, розробці базової архітектури та проведенні тестування прототипу.

Метою роботи є розробка системи управління контентом (CMS), орієнтованої на потреби малого бізнесу, що забезпечить простоту використання, адаптивність, безпеку та підтримку ключових функціональних можливостей.

Для досягнення цієї мети було визначено такі **завдання**:

- Провести аналіз існуючих рішень на ринку CMS та визначити їхні переваги та недоліки.
- Сформулювати технічні вимоги до розроблюваної системи, враховуючи потреби малого бізнесу.
- Розробити архітектуру системи з урахуванням нефункціональних вимог (безпека, гнучкість, масштабованість).
- Реалізувати функціональні можливості, включаючи модулі для роботи з контентом, SEO-оптимізацію та багатомовність.
- Провести тестування системи, оцінити її продуктивність та відповідність заданим вимогам.
- Розробити рекомендації щодо впровадження та використання системи.

Об'єктом дослідження є процес розробки та впровадження інформаційних систем для управління контентом малого бізнесу.

Предметом дослідження є функціональні, технічні та організаційні елементи створення CMS, орієнтованої на малий бізнес, із врахуванням сучасних технологій та вимог користувачів.

Практичне значення отриманих результатів. Розроблена система управління контентом може бути використана малими підприємствами для створення та адміністрування вебсайтів без значних фінансових та технічних затрат. Запропоновані рішення сприяють підвищенню ефективності комунікації з клієнтами та оптимізації внутрішніх бізнес-процесів.

Система може бути впроваджена як на локальних серверах, так і в хмарних середовищах, що робить її гнучкою у використанні. Результати дослідження можуть бути використані для подальшої розробки адаптивних рішень у галузі управління контентом, а також інтегровані у навчальні курси, присвячені розробці інформаційних систем.

Апробація результатів дослідження. Результати дослідження було представлено на засіданнях кафедри та групових зборах.

Під час апробації обговорювалися ключові елементи функціоналу розробленої системи, її переваги та можливості впровадження. Автор брав участь у формулюванні висновків, удосконаленні архітектури та розробці рекомендацій щодо впровадження системи.

Результати також було висвітлено у вигляді тез та доповідей на наукових семінарах, організованих кафедрою у 2024 році.

РОЗДІЛ 1. АНАЛІЗ ПОТРЕБ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

1.1. Аналіз потреб малого бізнесу в управлінні контентом

Малий бізнес є важливою складовою сучасної економіки, оскільки забезпечує створення нових робочих місць, сприяє розвитку локальних громад і задовольняє широкий спектр споживчих потреб. Однак для досягнення успіху в умовах жорсткої конкуренції підприємства повинні ефективно взаємодіяти зі своїми клієнтами, забезпечуючи оперативну, якісну та релевантну інформацію. Управління контентом є ключовим елементом цієї взаємодії, що робить використання сучасної системи управління контентом (CMS) нагальною потребою для малих підприємств [1].

Ефективна система управління контентом допомагає малому бізнесу вирішувати низку критичних завдань: від презентації товарів і послуг до організації зворотного зв'язку з клієнтами [2]. Для більшості малих підприємств, які часто мають обмежені ресурси, зокрема фінансові та людські, важливо, щоб CMS була зручною у використанні, доступною та функціональною [3]. Потреби в управлінні контентом притаманні малому бізнесу показані на рисунку 1.1 [4].



Рисунок 1.1. Потреби в управлінні контентом притаманні малому бізнесу

Інформаційна доступність для клієнтів: ключовим завданням для малого бізнесу є надання актуальної інформації про свої товари, послуги та спеціальні пропозиції [5]. Для цього необхідно, щоб CMS забезпечувала швидке додавання, редагування та видалення контенту. Наприклад, магазин може оперативно змінювати ціни на свої товари або додавати інформацію про нові послуги. Це не лише підвищує задоволеність клієнтів, але й дозволяє бізнесу залишатися конкурентоспроможним.

Також варто відзначити, що інформація повинна бути подана у зручному та зрозумілому вигляді, зокрема через текст, зображення та відео. Тому CMS повинна підтримувати різні формати мультимедійного контенту, дозволяючи бізнесу ефективно презентувати свою продукцію [6].

Інструменти для взаємодії з клієнтами: клієнти малого бізнесу часто потребують швидкого зворотного зв'язку [7]. Вбудована форма зворотного

зв'язку, доступна в CMS, допомагає встановити ефективну комунікацію між підприємством і його аудиторією [8]. Наприклад, клієнти можуть ставити запитання щодо продуктів, залишати відгуки або надсилати пропозиції.

Окрім форми зворотного зв'язку, важливим є налаштування інтеграції з соціальними мережами, оскільки багато клієнтів вважають за краще спілкуватися саме через ці платформи [9]. CMS, що пропонує такі інструменти, сприяє зростанню довіри до бізнесу та збільшенню кількості потенційних клієнтів.

Адаптивність і мобільність: з огляду на те, що сучасні користувачі дедалі частіше переглядають веб-сторінки з мобільних пристроїв, адаптивний дизайн стає критично важливим для малого бізнесу [10]. Сайт, який коректно відображається на смартфонах і планшетах, забезпечує кращий користувацький досвід, що позитивно впливає на поведінку клієнтів і сприяє зростанню продажів [11].

CMS повинна автоматично адаптувати контент під різні розміри екранів, а також дозволяти налаштовувати мобільні версії сторінок [12]. Для малого бізнесу це означає можливість охопити ширшу аудиторію без необхідності значних фінансових вкладень у розробку окремого мобільного сайту.

Безпека даних: захист інформації є одним із найважливіших елементів управління контентом [13]. Малий бізнес, який працює з персональними даними клієнтів, має забезпечити їхню конфіденційність і захист від несанкціонованого доступу. CMS повинна мати вбудовані інструменти безпеки, такі як шифрування даних, система резервного копіювання та захист від атак [14].

Особливо актуальним є питання безпеки в умовах хмарного розгортання, яке часто обирається малим бізнесом через його економічність та простоту в управлінні. Надійна CMS забезпечує захист як у локальному середовищі, так і в хмарі [15].

Простота у використанні: малий бізнес рідко має можливість утримувати спеціалізований відділ IT [16]. Тому CMS повинна бути простою та інтуїтивно

зрозумілою, щоб навіть люди без технічної підготовки могли легко працювати з нею. Це стосується як створення нових сторінок, так і управління існуючим контентом. Наприклад, інтерфейс адміністратора має бути логічно структурованим, а для виконання базових операцій не повинно знадобитися складного навчання.

Можливість масштабування: хоча малий бізнес зазвичай починає з обмежених обсягів контенту, успішний розвиток може потребувати розширення можливостей CMS[17]. Наприклад, зростання бізнесу може викликати необхідність додавання нових мов, розширення функціональності або інтеграції з іншими сервісами, такими як платіжні системи чи аналітичні платформи. Тому важливо, щоб CMS могла масштабуватися відповідно до потреб бізнесу.

Інструменти SEO та аналітики: для малого бізнесу, який прагне бути помітним в інтернеті, надзвичайно важливою є базова SEO-оптимізація [18]. CMS повинна підтримувати створення дружніх URL, редагування мета-тегів і генерацію XML-карт сайту. Такі можливості допомагають сайту бути вищим у пошукових запитах і привертати більше відвідувачів.

Крім того, інструменти для інтеграції з Google Analytics або іншими аналітичними платформами дозволяють власникам бізнесу відстежувати поведінку користувачів, визначати популярні сторінки та оцінювати ефективність маркетингових кампаній [19].

Отже, потреби малого бізнесу в управлінні контентом є багатограними, і їх ефективно задоволення можливе лише за допомогою сучасної CMS. Вона має бути простою у використанні, безпечною, адаптивною, а також підтримувати базову SEO-оптимізацію, мультимедійний контент і інструменти для зворотного зв'язку. Реалізація таких можливостей дозволяє малому бізнесу не лише покращити взаємодію з клієнтами, а й закріпити свої позиції на ринку.

1.2. Аналіз існуючих рішень

Сучасний ринок програмних рішень пропонує широкий спектр систем управління контентом (Content Management Systems, CMS), які адаптовані під різні завдання [20]. Вони варіюються за функціональністю, складністю впровадження, налаштуваннями, можливостями розширення та підтримкою специфічних потреб користувачів. Огляд найбільш популярних та затребуваних рішень дозволяє визначити їх переваги та недоліки, що стане основою для обґрунтування вибору технологій для розробки власної CMS.

1.2.1 WordPress

WordPress (рисунок 1.2) є однією з найпопулярніших платформ CMS у світі [21]. Початково створена як система для блогів, вона розвинулася до багатофункціональної платформи, що підтримує розробку сайтів різного рівня складності. WordPress є відкритим програмним забезпеченням і має величезну спільноту розробників та користувачів.

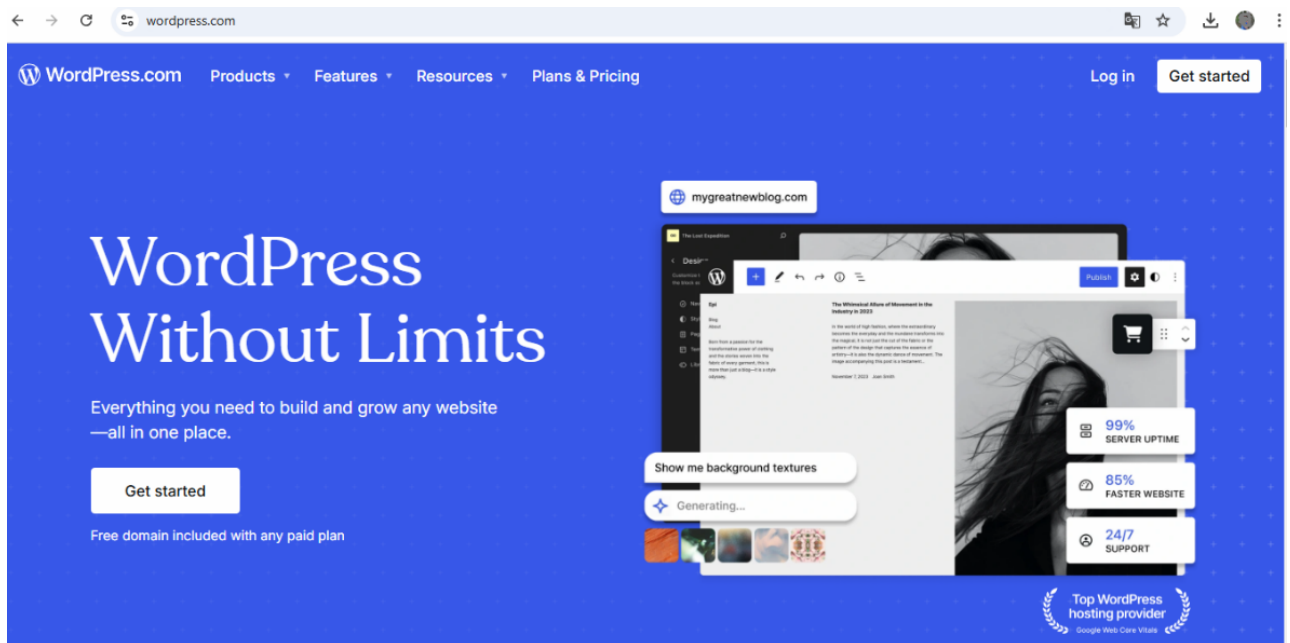


Рисунок 1.2. Головна сторінка WordPress

Функціонал:

- Додавання, редагування та видалення сторінок і записів.
- Підтримка текстів, зображень, відео та інших мультимедійних файлів.
- Інтеграція із широким спектром плагінів для розширення функціоналу.
- Наявність тем для створення унікального дизайну.
- Базова SEO-оптимізація (мета-теги, структура URL).
- Можливість створення мультимовних сайтів за допомогою додаткових плагінів.

Переваги WordPress подані в таблиці 1.1 [22, 23].

Таблиця 1.1

Переваги WordPress

№	Переваги	Опис
1	Легкість використання	Інтуїтивно зрозумілий інтерфейс адміністративної панелі забезпечує зручність для користувачів навіть без технічного досвіду
2	Масштабованість	Система дозволяє розширювати функціонал завдяки тисячам плагінів і тем, доступних у каталозі
3	SEO-оптимізація	WordPress надає інструменти для роботи з мета-тегами, ключовими словами, а також дружніми URL
4	Спільнота розробників	Велика кількість користувачів сприяє швидкому розвитку платформи, а також надає доступ до безлічі навчальних ресурсів

Недоліки WordPress подані в таблиці 1.2 [22, 23].

Таблиця 1.2

Недоліки WordPress

№	Недоліки	Опис
1	Безпека	Велика популярність робить платформу вразливою до атак, якщо не застосовуються регулярні оновлення та відповідні заходи безпеки
2	Навантаження	Велика кількість плагінів може призводити до значного споживання серверних ресурсів
3	Обмеження кастомізації	Для виконання складних завдань можуть знадобитися поглиблені знання програмування

WordPress підходить для невеликих бізнесів, які потребують швидкого запуску сайту з базовими функціями [24]. Однак для забезпечення безпеки та швидкості необхідні додаткові ресурси.

1.2.2. Joomla

Joomla (рисунок 1.3) — це CMS з відкритим кодом, яка забезпечує більшу функціональність і гнучкість у порівнянні з WordPress, але вимагає більшого рівня технічних знань [25].



Рисунок 1.3. Головна сторінка Joomla

Функціонал:

- Розширені можливості створення динамічного контенту.
- Вбудовані інструменти SEO.
- Підтримка багатомовності «з коробки».
- Інтеграція з каталогами товарів та послуг через компоненти, такі як

VirtueMart.

Переваги Joomla подані в таблиці 1.3 [26].

Таблиця 1.3

Переваги Joomla

№	Переваги	Опис
1	Гнучкість	Підходить для створення багатомовних сайтів і платформ із динамічним контентом
2	Розширена функціональність	Пропонує більше можливостей для створення комплексних проектів порівняно з WordPress
3	Модульна структура	Дозволяє легко додавати або змінювати функціонал

Недоліки Joomla подані в таблиці 1.4 [27].

Таблиця 1.4

Недоліки Joomla

№	Недоліки	Опис
1	Складність освоєння	Для новачків інтерфейс адміністративної панелі може здатися заплутаним
2	Менша кількість плагінів і шаблонів	У порівнянні з WordPress, вибір додаткових компонентів більш обмежений

Joomla ідеально підходить для сайтів із середньою складністю, але її складність може стати бар'єром для користувачів без технічного досвіду [28].

1.2.3. Drupal

Drupal (рисунок 1.4) є потужною платформою для розробки сайтів, що потребують високого рівня кастомізації [29]. Це одна з найгнучкіших CMS, яка підходить для створення складних проектів.

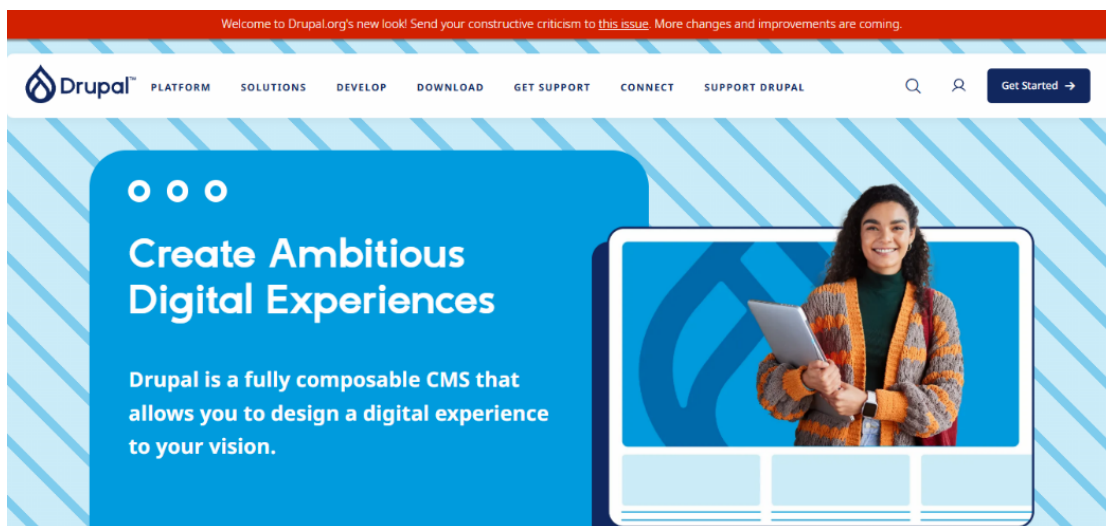


Рисунок 1.4. Головна сторінка Drupal

Функціонал:

- Гнучке управління користувачами та ролями.
- Розширені можливості для роботи з великим обсягом даних.
- Потужний API для інтеграції сторонніх сервісів.
- Багатомовність підтримується на рівні ядра.

Переваги Drupal подані в таблиці 1.5 [30].

Таблиця 1.5

Переваги Drupal

№	Переваги	Опис
1	Безпека	Високий рівень захисту робить платформу популярною серед урядових організацій та великих корпорацій
2	Гнучкість	Система дозволяє створювати сайти будь-якої складності завдяки модульній архітектурі
3	Широкі можливості кастомізації	Розробники можуть змінювати майже всі елементи системи

Недоліки Drupal подані в таблиці 1.6 [31].

Таблиця 1.6

Недоліки Drupal

№	Недоліки	Опис
1	Складність розробки	Для роботи із платформою потрібен значний рівень технічної підготовки
2	Висока вартість	Створення та підтримка сайтів на основі Drupal можуть бути затратними

Drupal краще використовувати для складних і масштабних проєктів [32]. Для малого бізнесу може бути надмірно складною.

1.2.4. Wix

Wix (рисунок 1.5) — це платформа для створення сайтів за принципом «перетягни і кинь» [33]. Вона є однією з найбільш зручних для користувачів без технічного досвіду.

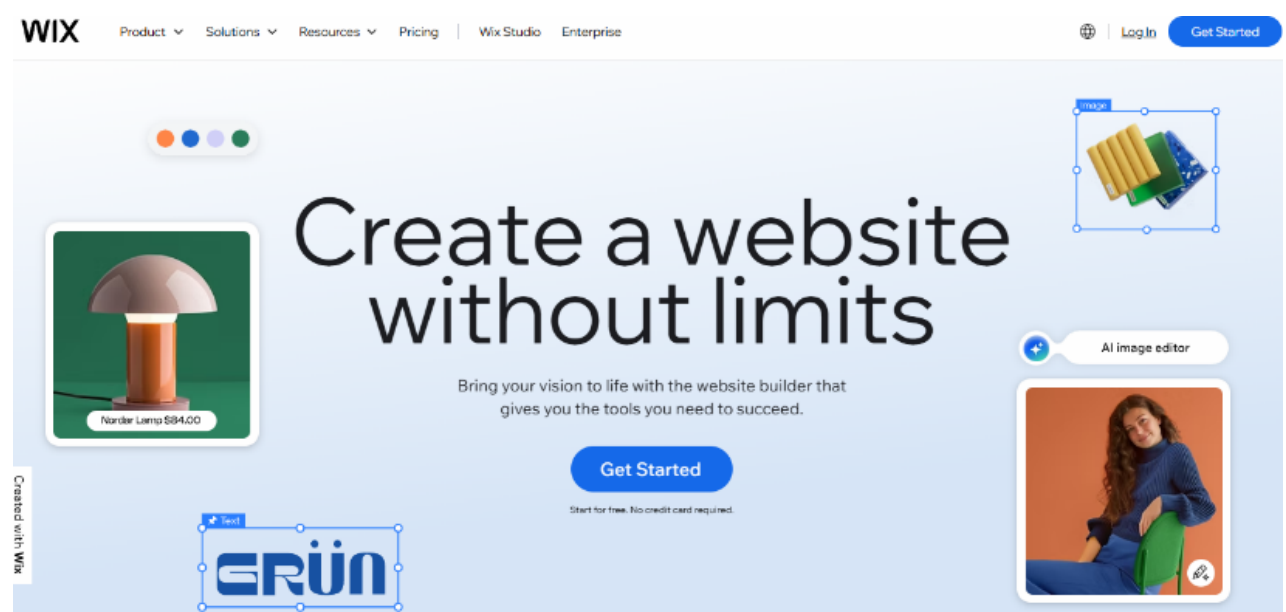


Рисунок 1.5. Головна сторінка Wix

Функціонал:

- Інтуїтивний конструктор для створення сторінок.
- Вбудовані SEO-інструменти.
- Публікація мультимедійного контенту.
- Базова аналітика через внутрішні інструменти.

Переваги Wix подані в таблиці 1.7 [34].

Таблиця 1.7

Переваги Wix

№	Переваги	Опис
1	Візуальний редактор	Інтуїтивний конструктор сайтів дозволяє створювати сторінки без навичок програмування
2	Шаблони	Платформа пропонує широкий вибір дизайнів для різних сфер бізнесу
3	Хостинг у комплекті	Wix не потребує окремої установки чи налаштування серверів

Недоліки Wix подані в таблиці 1.8 [35].

Таблиця 1.8

Недоліки Wix

№	Недоліки	Опис
1	Обмежена масштабованість	Для складних проектів платформа може виявитися недостатньою
2	Залежність від хмарної моделі	Відсутність можливості локального розгортання обмежує варіанти використання

Wix підходить для невеликих компаній, які хочуть швидко запустити сайт, не залучаючи розробників [36].

1.2.5. OpenCart

OpenCart (рисунок 1.6) — це CMS, орієнтована на створення інтернет-магазинів із середнім обсягом продажів [37].

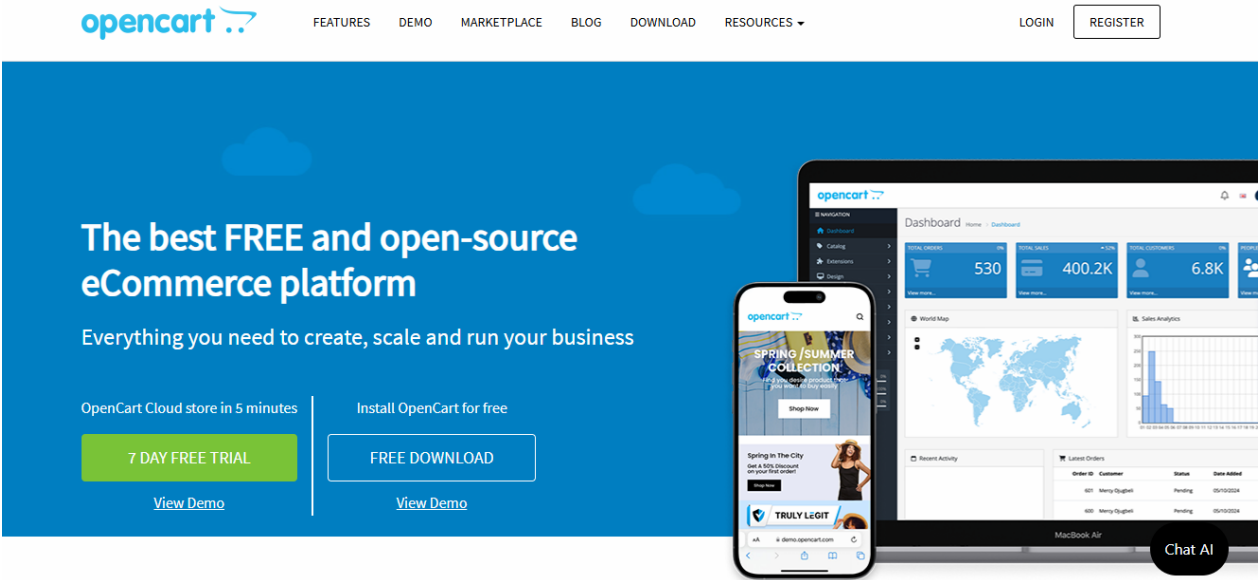


Рисунок 1.6. Головна сторінка OpenCart

Функціонал:

- Створення та управління товарами та категоріями.
- Підтримка мультивалютності та мультимовності.
- Інтеграція з платіжними шлюзами.

Переваги OpenCart подані в таблиці 1.9 [38].

Таблиця 1.9

Переваги OpenCart

№	Переваги	Опис
1	Легкість адміністрування	Інтерфейс платформи оптимізований для роботи з товарами та замовленнями
2	Модульна структура	Дозволяє додавати нові функції за допомогою модулів
3	Дружні ціни	Використання платформи не вимагає значних фінансових вкладень

Недоліки OpenCart подані в таблиці 1.10 [38].

Таблиця 1.10

Недоліки OpenCart

№	Недоліки	Опис
1	Обмежена підтримка великих каталогів	Платформа може бути неефективною для сайтів із великим асортиментом товарів
2	Складності в SEO	OpenCart потребує додаткових налаштувань для забезпечення конкурентоспроможної SEO-оптимізації

OpenCart є хорошим вибором для малого бізнесу з невеликим обсягом продажів [39].

1.2.6. Shopify

Shopify (рисунок 1.7) — це хмарна CMS, яка спеціалізується на створенні інтернет-магазинів і забезпечує широкий функціонал для електронної комерції [40].

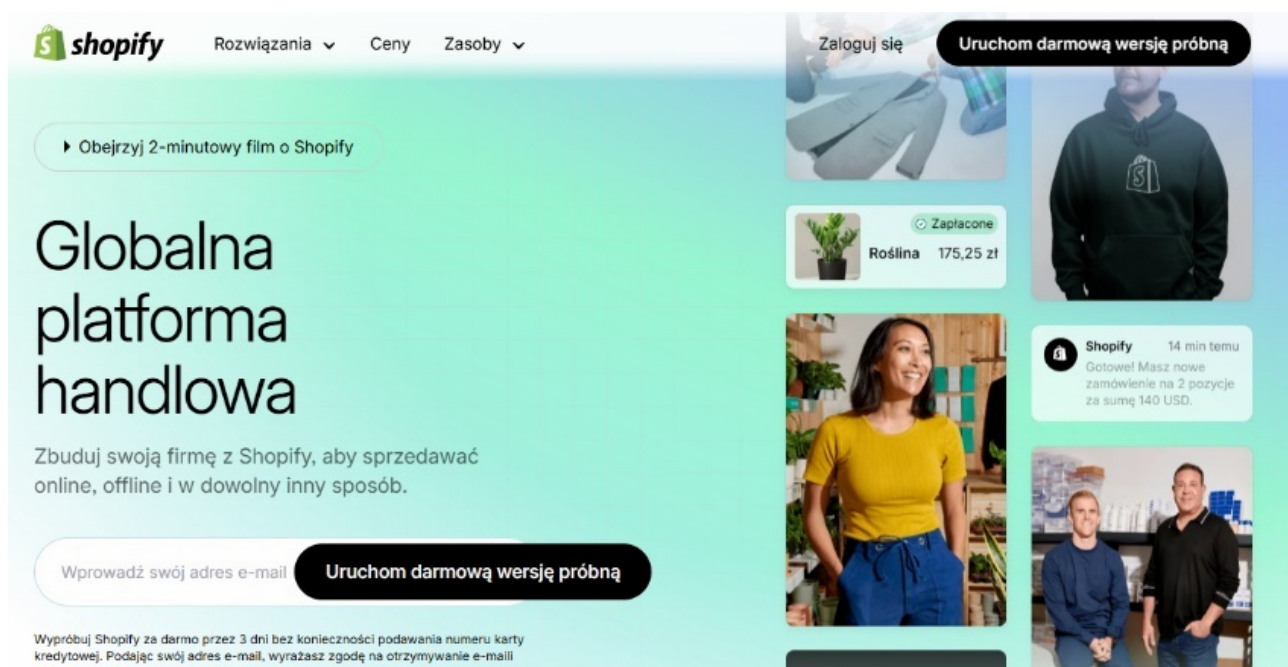


Рисунок 1.7. Головна сторінка Shopify

Функціонал:

- Повний набір інструментів для управління товарами, замовленнями та клієнтами.
- Інтеграція з платіжними системами та підтримка мультивалютності.
- Гнучкі налаштування дизайну через конструктор шаблонів.
- Вбудовані інструменти для SEO та маркетингу (знижки, промокоди, email-розсилки).
- Інтеграція з платформами аналітики, такими як Google Analytics.

Переваги Shopify подані в таблиці 1.11 [41].

Таблиця 1.11

Переваги Shopify

№	Переваги	Опис
1	Простота використання	Інтуїтивно зрозумілий інтерфейс дозволяє швидко налаштовувати магазин без залучення розробників
2	Велика екосистема додатків	Доступні численні розширення для інтеграції з CRM, логістичними службами тощо
3	Стабільність і швидкість роботи	Хмарна інфраструктура забезпечує високу продуктивність і надійність
4	Підтримка	Цілодобова технічна підтримка для користувачів

Недоліки Shopify подані в таблиці 1.12 [42].

Таблиця 1.12

Недоліки Shopify

№	Недоліки	Опис
1	Вартість	Високі щомісячні тарифи та комісії за транзакції
2	Обмеження кастомізації	Дизайнерські можливості обмежені в порівнянні з деякими іншими CMS
3	Залежність від платформи	Неможливість перенесення сайту на інші сервери

Shopify є оптимальним рішенням для бізнесів, що займаються електронною комерцією, особливо якщо потрібен швидкий старт і мінімальні технічні складнощі [43].

1.2.7. MODX

MODX (рисунок 1.8) — це універсальна CMS із відкритим кодом, яка надає розробникам повний контроль над створенням та управлінням вебсайтами [44].

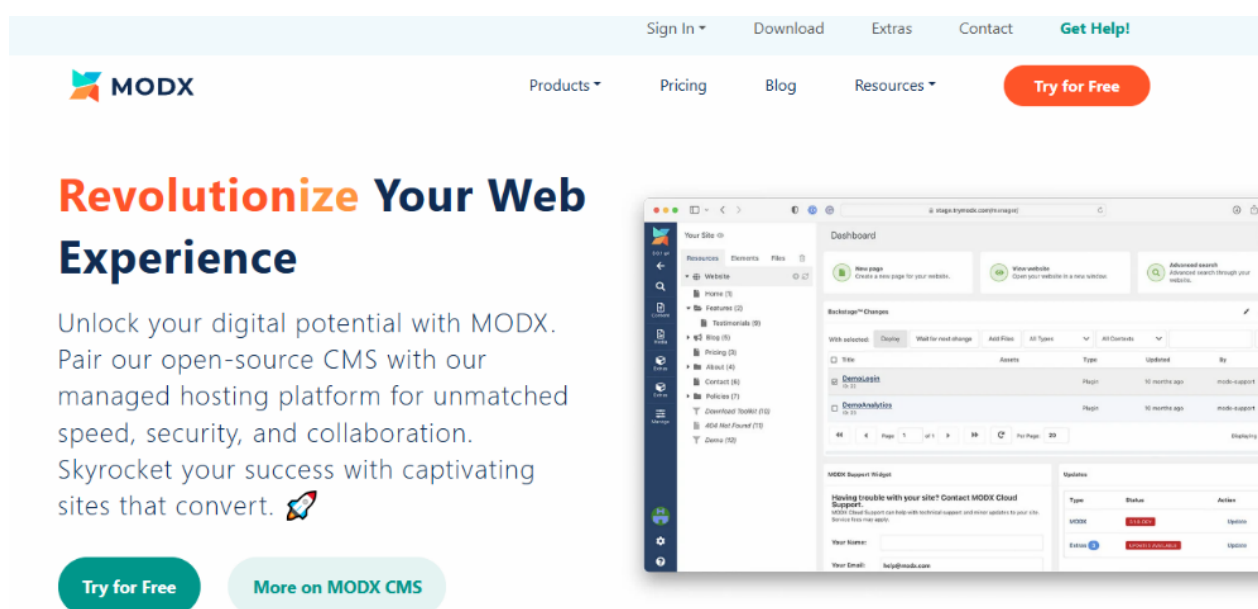


Рисунок 1.8. Головна сторінка MODX

Функціонал:

- Потужний редактор для управління структурою та контентом.
- Розширені можливості для кастомізації за допомогою шаблонів і модулів.
- Інтеграція з зовнішніми API для розширення функціоналу.
- Вбудована SEO-оптимізація (створення дружніх URL, редагування мета-тегів).
- Підтримка мультимовності.

Переваги MODX подані в таблиці 1.13 [45].

Таблиця 1.13

Переваги MODX

№	Переваги	Опис
1	Гнучкість у налаштуванні	Можливість створювати унікальні проекти з нестандартною логікою
2	Безпека	Часті оновлення та захист від поширених атак
3	Відсутність залежності від шаблонів	Платформа дозволяє реалізовувати будь-який дизайн або функціонал

Недоліки MODX подані в таблиці 1.14 [46].

Таблиця 1.14

Недоліки MODX

№	Недоліки	Опис
1	Складність освоєння	Вимагає від адміністратора та розробника значних технічних знань
2	Обмежена спільнота	У порівнянні з популярнішими CMS, підтримка спільноти менша
3	Ресурсозатратність	Для масштабних проектів можуть знадобитися більше серверних ресурсів

MODX ідеально підходить для проектів, що потребують нестандартних рішень і високого рівня кастомізації, але може бути занадто складною для невеликих бізнесів без технічного персоналу [47].

Кожна з розглянутих платформ має свої сильні сторони та обмеження. Вибір CMS залежить від специфіки бізнесу, бюджету, технічного рівня команди, а також від очікувань щодо функціональності системи. У контексті розробки системи управління контентом для малого бізнесу особливу увагу слід приділити простоті використання, безпеці та можливості адаптації до зростання компанії. Це вимагає вибору інструментів, які поєднують у собі легкість адміністрування, базову SEO-оптимізацію, а також підтримку мобільних пристроїв та інтеграцію з зовнішніми сервісами.

1.3. Обґрунтування вибору технологій для розробки CMS

Розробка системи управління контентом (CMS) для малого бізнесу вимагає ретельного підходу до вибору технологій, оскільки її функціональні можливості, продуктивність, зручність використання та безпека мають бути відповідно адаптовані до потреб кінцевих користувачів. Основним завданням є створення такої системи, яка дозволить малим підприємствам ефективно управляти своїм контентом без необхідності значних інвестицій у навчання або технічну підтримку. Використані технології повинні забезпечувати гнучкість, масштабованість та безпеку, залишаючись водночас доступними для впровадження.

Основою створення такої CMS стали сучасні веб-технології, зокрема Python, Django, HTML, CSS, JavaScript і сторонні бібліотеки, що полегшують інтеграцію з необхідними сервісами та підвищують зручність розробки [48]. Обрання цих технологій має низку обґрунтованих причин.

Перш за все, вибір Python як основної мови програмування зумовлений її універсальністю, читабельністю та широким спектром бібліотек і фреймворків. Python дозволяє розробляти складні веб-додатки з мінімальними витратами часу і ресурсів завдяки своєму лаконічному синтаксису [49]. Django, який використовується як основний фреймворк, забезпечує надійну архітектуру для створення веб-додатків, надаючи розробникам можливість працювати з уже готовими інструментами для аутентифікації, маршрутизації, управління базами даних, а також підтримки багатомовності. Завдяки вбудованому адміністраторському інтерфейсу Django спрощує роботу з контентом і дозволяє користувачам легко управляти сторінками, формами та іншими елементами.

HTML, CSS і JavaScript забезпечують створення сучасного користувацького інтерфейсу з адаптивним дизайном [50]. HTML використовується для структурування контенту, CSS дозволяє стилізувати сторінки, роблячи їх привабливими та функціональними, а JavaScript додає

інтерактивності, зокрема динамічну зміну контенту та інтеграцію сторонніх сервісів. Використання фреймворку Bootstrap, який поєднує CSS і JavaScript, дозволяє створювати адаптивний дизайн, що є критично важливим для малого бізнесу, оскільки значна частина користувачів переглядає сайти через мобільні пристрої.

Що стосується бази даних, то її роль виконує SQLite. Вона забезпечує простоту в розгортанні і використанні, що відповідає вимогам малого бізнесу, де ресурсів на управління складними системами може бути недостатньо. SQLite є вбудованим рішенням у Django, що дозволяє скоротити час на налаштування інфраструктури і спростити процес розробки [51]. Водночас ця база даних забезпечує достатній рівень продуктивності для невеликих сайтів і додатків, а її невеликі системні вимоги сприяють економії ресурсів сервера.

Для підвищення рівня безпеки системи використано механізми, надані Django, включно із захистом від міжсайтових атак, ін'єкцій SQL та інших поширених загроз. Аутентифікація користувачів базується на вбудованій системі управління правами доступу, що дозволяє гнучко налаштовувати рівні доступу до окремих частин сайту або панелі адміністратора. Додатково інтегровані функції резервного копіювання даних, що дозволяє зберігати їх цілісність навіть у випадку технічних збоїв.

Одним із важливих елементів є підтримка багатомовності, яка реалізована за допомогою системи перекладів у Django. Цей механізм дозволяє створювати багатомовні сайти без необхідності значного переписування коду. Всі текстові дані, включно з мета-тегами та повідомленнями для користувачів, підтримують переклад за допомогою шаблонних тегів, що спрощує управління мовними версіями.

Для інтеграції аналітичних інструментів використовується Google Analytics [52]. Завдяки цьому власники малого бізнесу можуть отримувати докладну інформацію про відвідувачів сайту, що дозволяє оцінювати ефективність контенту та маркетингових стратегій. Скрипти інтеграції

додаються до шаблонів динамічно, що дозволяє адміністратору увімкнути або вимкнути їх через налаштування без необхідності змінювати код.

Особлива увага приділена SEO-оптимізації. Всі сторінки підтримують додавання мета-тегів, включно з заголовками, описами і ключовими словами. Це дозволяє поліпшити видимість сайту в пошукових системах, що є критично важливим для малого бізнесу, оскільки більшість потенційних клієнтів шукають послуги або товари онлайн. Структура URL створена таким чином, щоб вони були зрозумілими та дружніми до пошукових систем, що також позитивно впливає на SEO [53].

Зручність для користувачів стала основним орієнтиром під час розробки інтерфейсу. Адміністративна панель максимально спрощена, що дозволяє користувачам без технічних навичок швидко додавати або редагувати контент. Форми зворотного зв'язку реалізовані за допомогою вбудованих механізмів Django Forms, що забезпечує їх високу функціональність, а також дає змогу обробляти повідомлення безпосередньо через інтерфейс адміністратора [54].

Таким чином, обрані технології повністю відповідають заявленим технічним вимогам і потребам малого бізнесу. Вони дозволяють створити гнучку, надійну та зручну систему управління контентом, яка є легкою в розгортанні, економічною в обслуговуванні та забезпечує високий рівень безпеки і функціональності. Усі ці переваги роблять систему ідеальним вибором для малого бізнесу, який прагне підвищити свою присутність в інтернеті і покращити взаємодію з клієнтами.

Висновки до розділу 1

Аналіз потреб малого бізнесу в управлінні контентом і вибір технологій для створення CMS демонструють необхідність використання сучасних, ефективних інструментів, що забезпечують зручність для користувача. Малий бізнес потребує систем, які підтримують публікацію текстів, зображень і відео, адаптивність, базову SEO-оптимізацію, мультимовність і аналітичні сервіси. Це визначає важливість технологій, які забезпечують функціональність, безпеку та зручність.

Вибір технологій базувався на аналізі існуючих рішень, кожне з яких має свої переваги й обмеження. WordPress, Joomla чи Drupal пропонують широкий функціонал, але часто потребують додаткових налаштувань і не завжди відповідають потребам малого бізнесу щодо простоти та масштабованості. Тому було обрано Python і Django, які забезпечують гнучкість, швидкість розробки й адаптивність.

Django як фреймворк пропонує потужні інструменти для створення веб-додатків. Архітектура "Model-View-Template" забезпечує розділення даних, логіки й інтерфейсу, що спрощує розробку й масштабування. Висока безпека, включно із захистом від SQL-ін'єкцій і CSRF-атак, критично важлива для захисту даних користувачів. Використання Python і бібліотек спростило реалізацію редактора контенту, оптимізацію зображень і створення резервних копій.

CMS мала враховувати нефункціональні вимоги: швидкість, адаптивність і легкість адміністрування. Django дозволив реалізувати зручну адмінпанель, доступну навіть для користувачів без технічного досвіду, що знижує витрати на обслуговування системи.

SEO-інструменти, такі як дружні URL, мета-теги та автоматична карта сайту, покращують видимість у пошукових системах і допомагають залучати нових клієнтів. Локалізація контенту сприяє охопленню ширшої аудиторії.

Для мобільної доступності застосовано стандарти HTML5, CSS3 і фреймворк Bootstrap. Це забезпечило адаптацію інтерфейсу для різних екранів і швидкий дизайн без втрати якості.

SQLite обрано як просту та автономну базу даних, що ідеально підходить для малого бізнесу. Незважаючи на обмеження масштабованості, вона задовольняє базові потреби зберігання даних. Реалізований функціонал резервного копіювання у форматі JSON спрощує відновлення інформації.

Таким чином, обрані технології дозволили створити багатофункціональну CMS, яка відповідає сучасним стандартам розробки, задовольняє потреби малого бізнесу й залишається відкритою для подальшого розширення. Django й Python забезпечили стабільність, безпеку та зручність системи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ

2.1. Функціональні вимоги до CMS

Ефективна система управління контентом (CMS) є невід'ємною частиною сучасного малого бізнесу, оскільки дозволяє компаніям створювати, редагувати та публікувати контент із мінімальними зусиллями. З огляду на потреби сучасних підприємств, розробка CMS вимагає інтеграції широкого спектру функцій, спрямованих на спрощення управління контентом, забезпечення зручності для користувачів та гнучкості в налаштуваннях. У межах цього дослідження визначено вимоги до CMS, зокрема функціональні можливості, які допомагають підприємцям у створенні динамічного вебресурсу, що відповідає їхнім бізнес-цілям.

Основні функції для управління контентом представлені на рисунку 2.1.



Рисунок 2.1. Функції управління контентом

Ключовою вимогою до будь-якої CMS є забезпечення можливості працювати зі сторінками сайту (додавання, редагування та видалення сторінок). У рамках реалізації цієї функції користувачі отримують інтерфейс, де вони можуть створювати нові сторінки, редагувати існуючі чи видаляти їх.

У реалізованому проєкті для адміністраторів передбачено зручний інтерфейс, який дозволяє:

- задавати заголовки сторінок;
- завантажувати зображення, додаючи їх до основного тексту;
- налаштовувати мета-теги, такі як meta-title, meta-description та meta-keywords, для кожної сторінки.

CMS має підтримувати можливість інтеграції текстів, зображень та відео (Публікація текстового, графічного та мультимедійного контенту). Адміністратор через простий візуальний редактор може форматовувати текст, вставляти зображення за допомогою спеціальних полів та завантажувати відео. Це дає змогу створювати контент, який виглядає професійно та відповідає сучасним стандартам дизайну.

Однією з важливих функцій CMS є форма зворотного зв'язку, яка дозволяє клієнтам зв'язуватися з бізнесом напряму через вебсайт. У розробленій системі передбачено форму, що містить поля для введення імені, електронної пошти та текстового повідомлення. Додатково реалізовано перевірку на заповнення обов'язкових полів та захист від помилкових даних, що підвищує якість роботи системи.

Система управління контентом підтримує багатомовність, забезпечуючи можливість адаптації вебсайту до різних мовних аудиторій. У розробленому проєкті передбачено використання бібліотеки i18n для динамічного перемикання мов. Користувачі можуть швидко вибирати бажану мову через інтерфейс, а вміст автоматично відображатиметься відповідно до вибору.

Інструменти для забезпечення функціональності показані на рисунку 2.2.



Рисунок 2.2. Інструменти для забезпечення функціональності

Простота інтерфейсу адміністратора є важливою складовою для малих бізнесів, де час і технічні ресурси часто обмежені. У створеній системі використано компонентний підхід до дизайну, де кожна функція представлена у вигляді окремого модуля. Наприклад, меню навігації, форми зворотного зв'язку та інші елементи адаптовані для зручної роботи навіть користувачами без досвіду програмування.

SEO-оптимізація впроваджена шляхом інтеграції мета-тегів, що дозволяє власникам бізнесу самостійно налаштовувати ключові слова, опис сторінки та заголовки. Це забезпечує кращу видимість сайту в пошукових системах.

Одна з додаткових функцій, що підвищує цінність CMS для малого бізнесу, — можливість створення каталогу товарів або послуг. У межах проекту адміністратори можуть додавати нові товари, редагувати їх описи та завантажувати зображення. Каталог інтегровано у вигляді динамічних сторінок із можливістю сортування.

Для відстеження ефективності вебсайту було додано функцію інтеграції з Google Analytics. Адміністратори можуть отримувати статистику про поведінку користувачів, перегляди сторінок і показники конверсій. Це сприяє прийняттю виважених рішень для поліпшення контенту.

CMS забезпечує високий рівень захисту даних, зокрема:

- використання механізму CSRF-token для запобігання атакам;
- захист форм від спаму через серверну валідацію;
- зберігання інформації у незв'язаній базі даних, що зменшує ризик компрометації.

Система дозволяє створювати резервні копії бази даних, щоб у разі технічних проблем користувачі могли відновити втрачену інформацію. Адміністратор має можливість завантажувати копії даних у зручному форматі.

Розроблена система управління контентом враховує сучасні потреби малого бізнесу, забезпечуючи зручність використання, функціональність і безпеку. Наявність адаптивного дизайну, інструментів для SEO, підтримки багатомовності та динамічного управління контентом гарантує ефективну роботу з ресурсом, що сприяє розвитку бізнесу та задоволенню потреб клієнтів.

2.2. Архітектура системи

Архітектура системи управління контентом (CMS) є основою для створення та підтримки структури контенту, що дозволяє ефективно керувати та публікувати інформацію в цифровому вигляді. Важливою засадою роботи є розгляд основних компонентів та етапів взаємодії системи, з урахуванням діаграм, моделей і структурних схем. Це дозволяє створити чітку картину взаємодії між елементами CMS, а також забезпечити її ефективність, безпеку та масштабованість.

Архітектура CMS складається з кількох ключових компонентів, кожен з яких виконує певні функції в системі. Ці компоненти повинні взаємодіяти таким чином, щоб забезпечити зручність для користувача та стабільність у

роботі системи. Необхідно розглянути основні компоненти та їх взаємодію через діаграми.

Користувачі системи можуть мати різні рівні доступу, які визначають їх права на доступ до функцій і контенту. Відповідно діаграми класів (рисунок 2.3), у системі визначаються три основні ролі користувачів: адміністратори, контент-менеджери та звичайні користувачі (відвідувачі). Це дає можливість точно налаштувати права доступу та здійснювати управління контентом на різних рівнях. У класах кожного користувача визначені відповідні атрибути та методи, які реалізують доступ до бази даних.

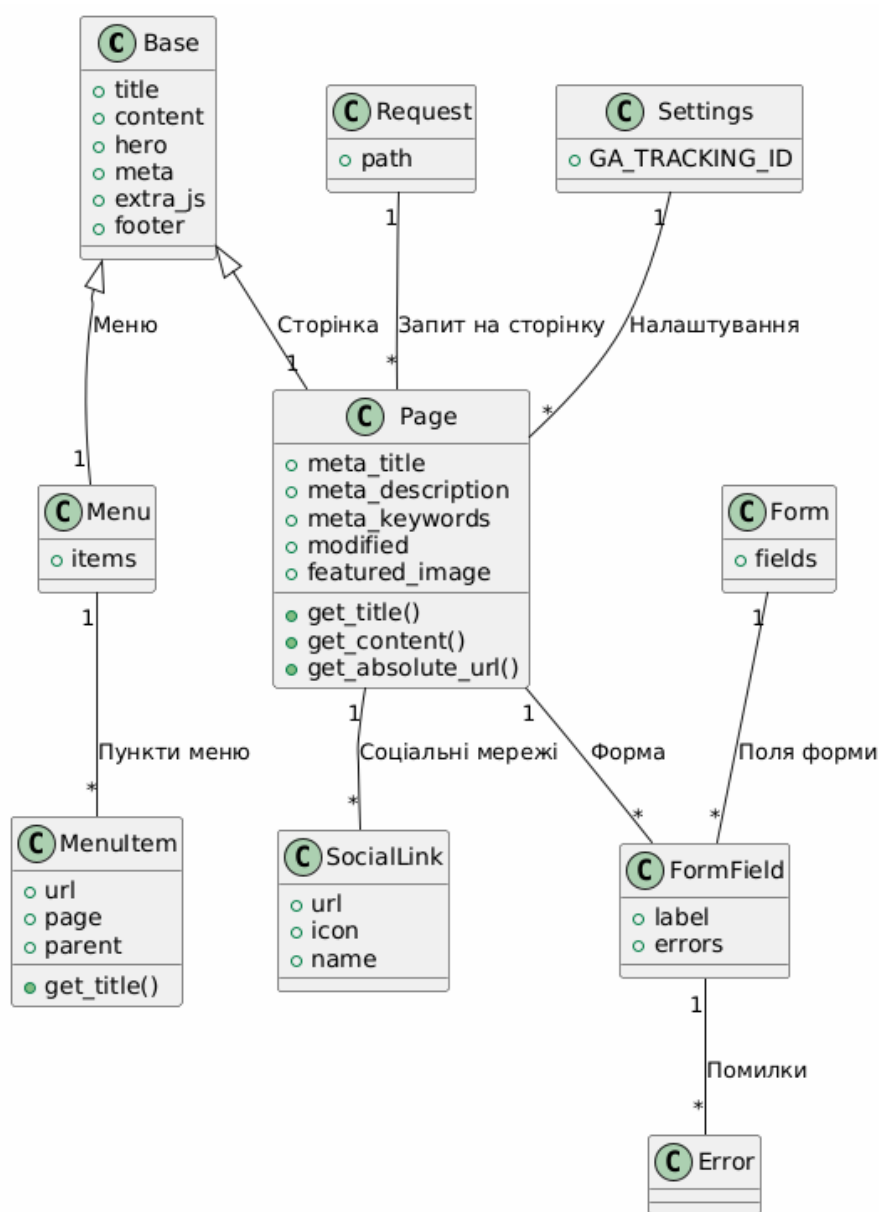


Рисунок 2.3. Діаграма класів

Діаграма класів детально показує об'єкти користувачів та їх взаємодії в системі: адміністратори мають повний доступ, контент-менеджери — доступ до створення та редагування контенту, а користувачі можуть лише переглядати публічний контент.

Додатково, діаграма варіантів використання (рисунок 2.4) дозволяє продемонструвати основні дії, які виконуються кожною роллю, наприклад створення нового контенту, редагування сторінки або перегляд публічного контенту. Це дає чітке уявлення про взаємодію різних ролей з системою та її функціональність.

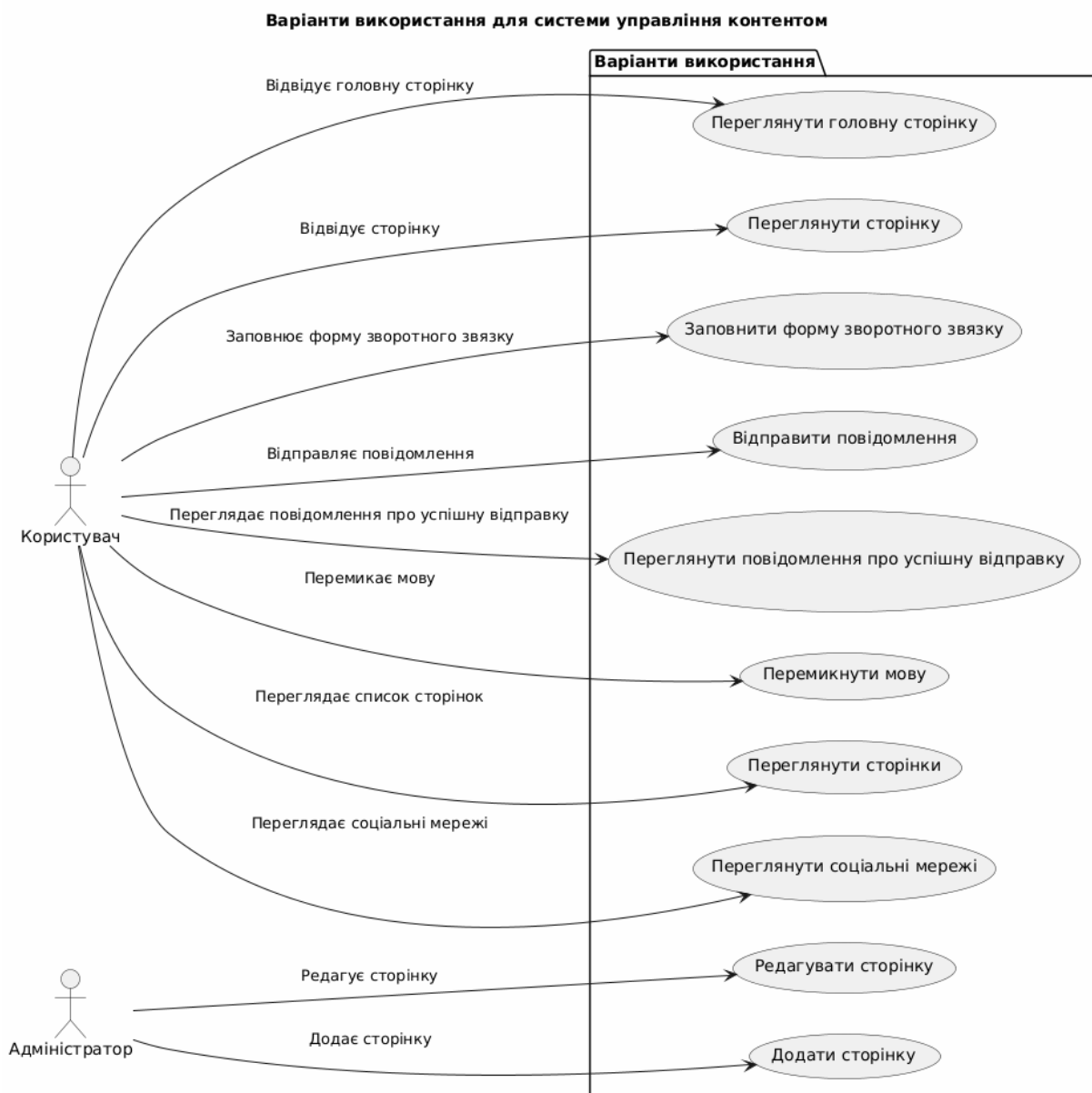


Рисунок 2.4. Діаграма варіантів використання

База даних Серце будь-якої CMS — це база даних, де зберігаються всі сторінки контенту, медіа, метадані та дані користувачів. Структура бази даних повинна бути чітко спроектована, щоб зберігання та доступ до даних були максимально ефективними. Структурна схема бази даних, як показано на рисунку 2.5, включає таблиці для зберігання:

- Інформації про сторінки контенту (назва, дата створення, автор тощо).
- Медіа-даних (зображення, відео, файли).
- Метаданих для кожної сторінки (опис, ключові слова).

діаграмою компонентів (рисунок 2.6), сервер складається з кількох основних модулів:

- Модуль аутентифікації та авторизації (забезпечує контроль доступу до системи).
- Модуль обробки контенту (забезпечує створення, редагування та видалення контенту).
- Модуль взаємодії з базою даних (зберігає та отримує необхідні дані з бази).

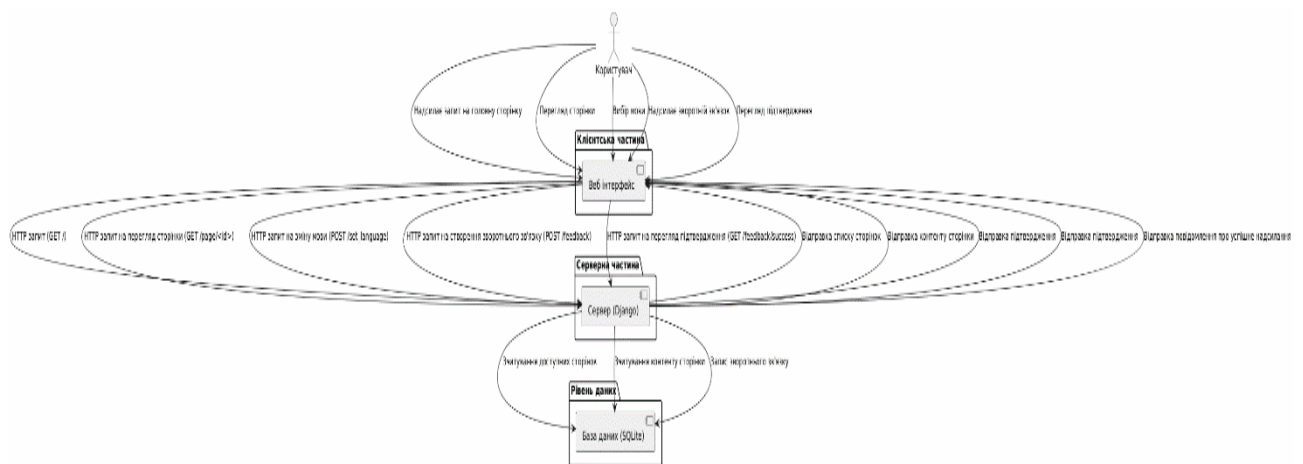


Рисунок 2.6. Діаграма компонентів

На кожному з етапів обробки запиту важливо, щоб система зберігала високу продуктивність при мінімальних затримках.

Діаграма діяльності (рисунок 2.7) чітко ілюструє ключові процеси у серверній частині, такі як отримання запиту, перевірка прав доступу, обробка даних у базі даних і повернення результатів користувачу. Це допомагає зрозуміти основні кроки та логіку роботи системи на рівні виконання функцій.

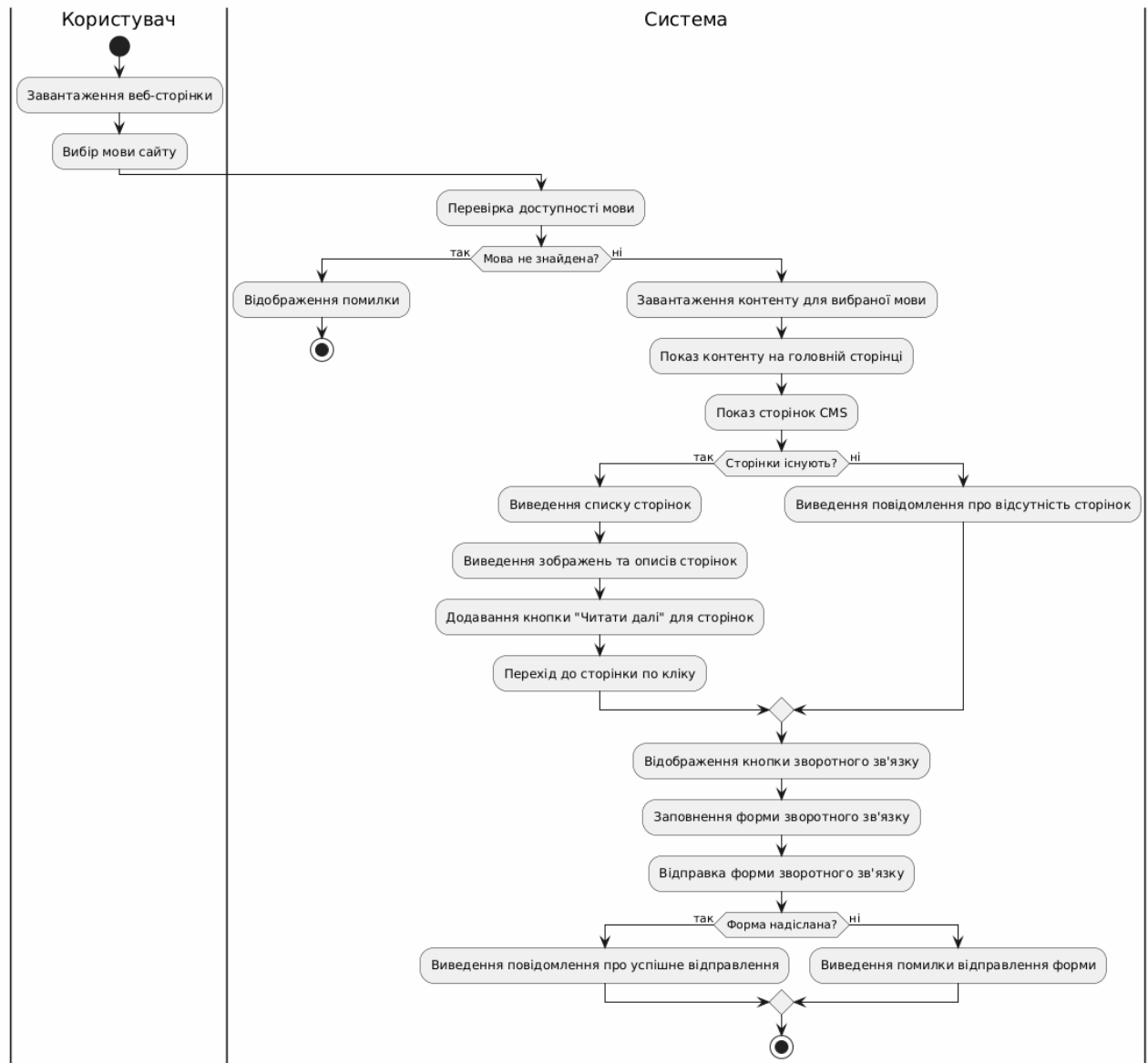


Рисунок 2.7. Діаграма діяльності

Клієнтська частина (Frontend) Клієнтська частина є інтерфейсом взаємодії користувача з CMS. Вона повинна бути зручною та інтуїтивно зрозумілою для різних типів користувачів. Згідно з діаграмою кооперацій (рисунок 2.8), клієнтська частина включає модулі:

- Модуль відображення контенту (відповідає за формування сторінок та відображення контенту).
- Модуль інтерактивності (відповідає за динамічні елементи інтерфейсу: форми зворотного зв'язку, коментарі, рейтинги).

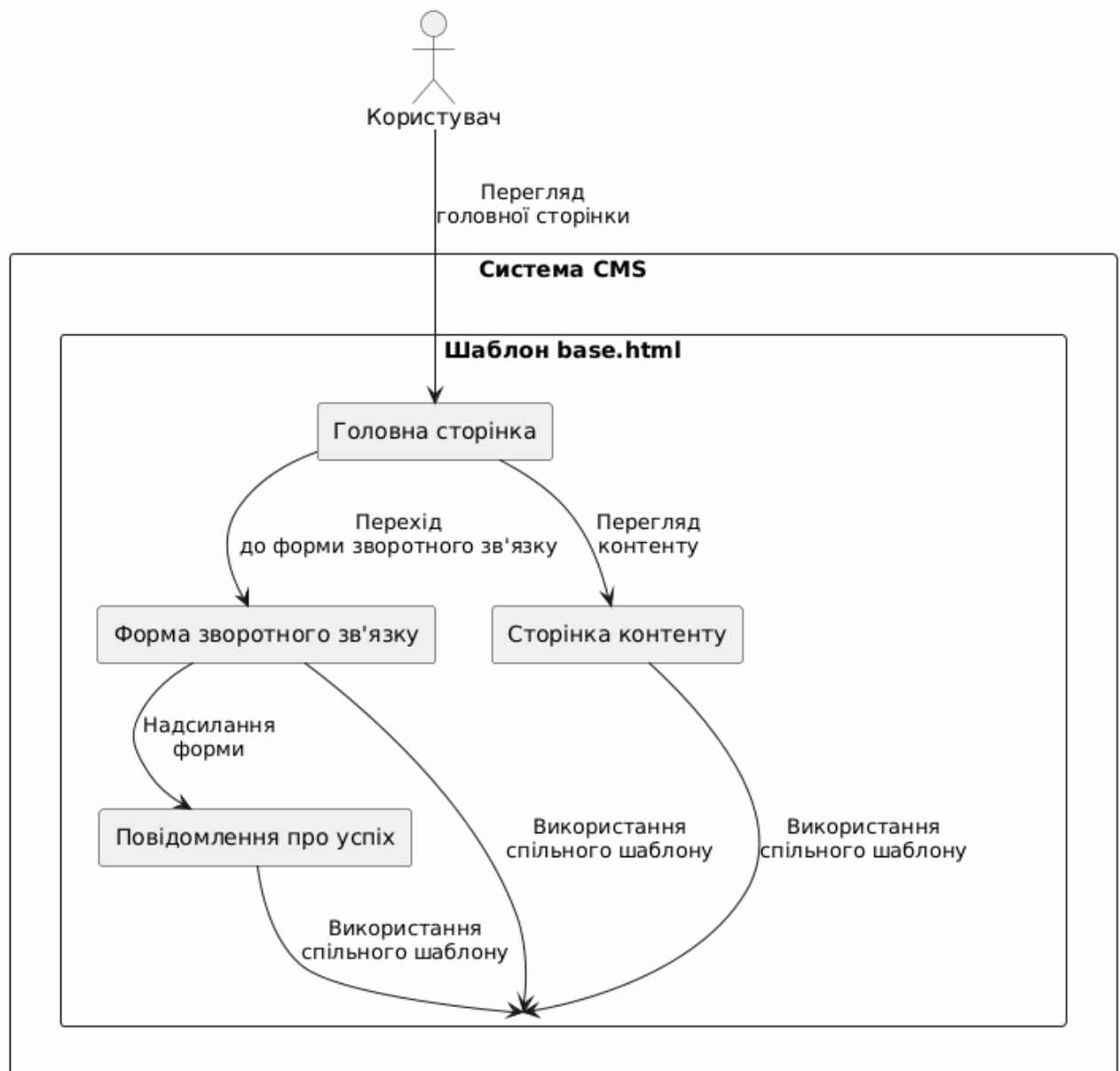


Рисунок 2.8. Діаграма кооперацій

Завдяки використанню таких технологій, як JavaScript та CSS, система забезпечує адаптивний дизайн та оптимізовану роботу на різних пристроях.

Шар безпеки Безпека є критично важливим елементом архітектури. Згідно з діаграмою станів (рисунок 2.9), кожен запит, що надходить до системи, проходить кілька етапів перевірки безпеки. Це включає:

- Шифрування паролів користувачів.
- Захист від SQL-ін'єкцій через використання підготовлених запитів.
- Аудит і контроль доступу для кожного користувача відповідно до його ролі.

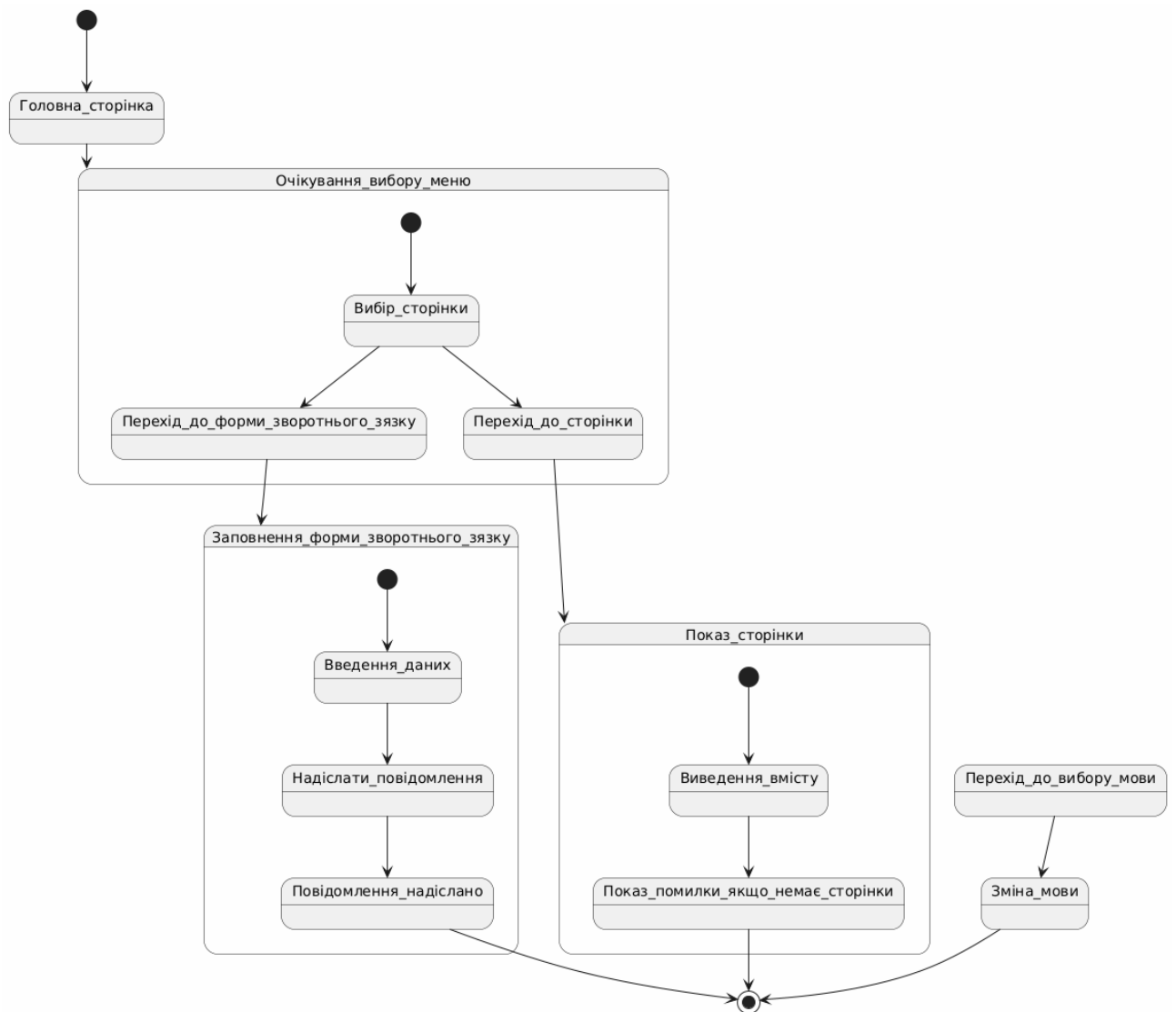


Рисунок 2.9. Діаграма станів

Важливим елементом успішної роботи CMS є правильна взаємодія між усіма компонентами. Це забезпечує коректне виконання всіх функцій системи, а також ефективне обслуговування запитів користувачів.

Запит від користувача: користувач відправляє запит на перегляд контенту, створення нової сторінки або редагування існуючої. Цей запит обробляється серверною частиною.

Обробка запиту на сервері: сервер приймає запит, перевіряє права користувача та взаємодіє з базою даних для отримання або оновлення необхідної інформації. Діаграма послідовностей (рисунок 2.10) показує

покрокове оброблення запиту: від отримання запиту до його виконання та передачі результатів користувачу.

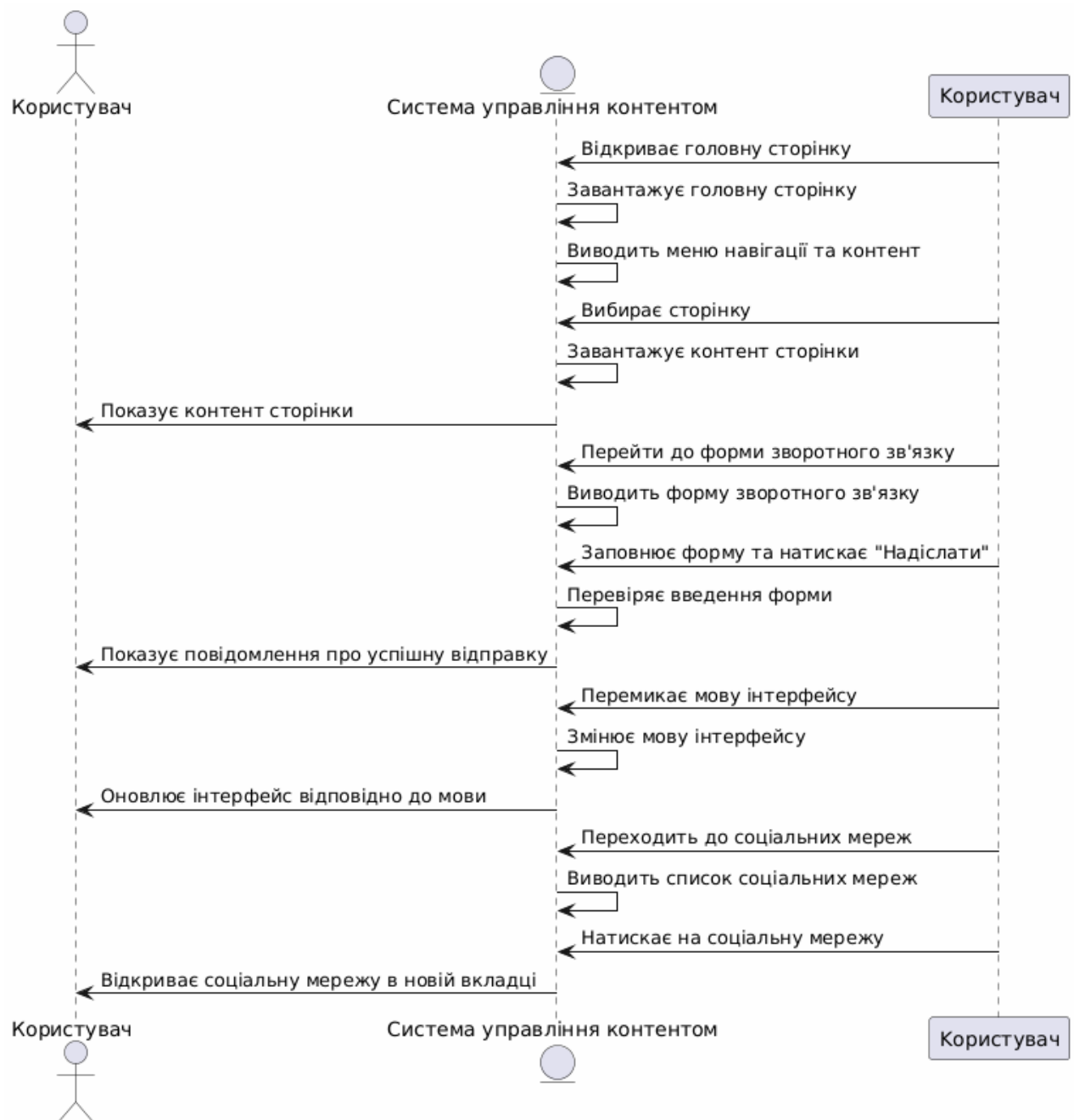


Рисунок 2.10. Діаграма послідовностей

Взаємодія з базою даних: якщо запит вимагає змін в даних (наприклад, редагування сторінки), сервер звертається до бази даних для оновлення або отримання потрібної інформації. Діаграма розгортання (рисунок 2.11) ілюструє,

як сервер взаємодіє з базою даних на рівні апаратного забезпечення та програмного забезпечення для виконання запитів.

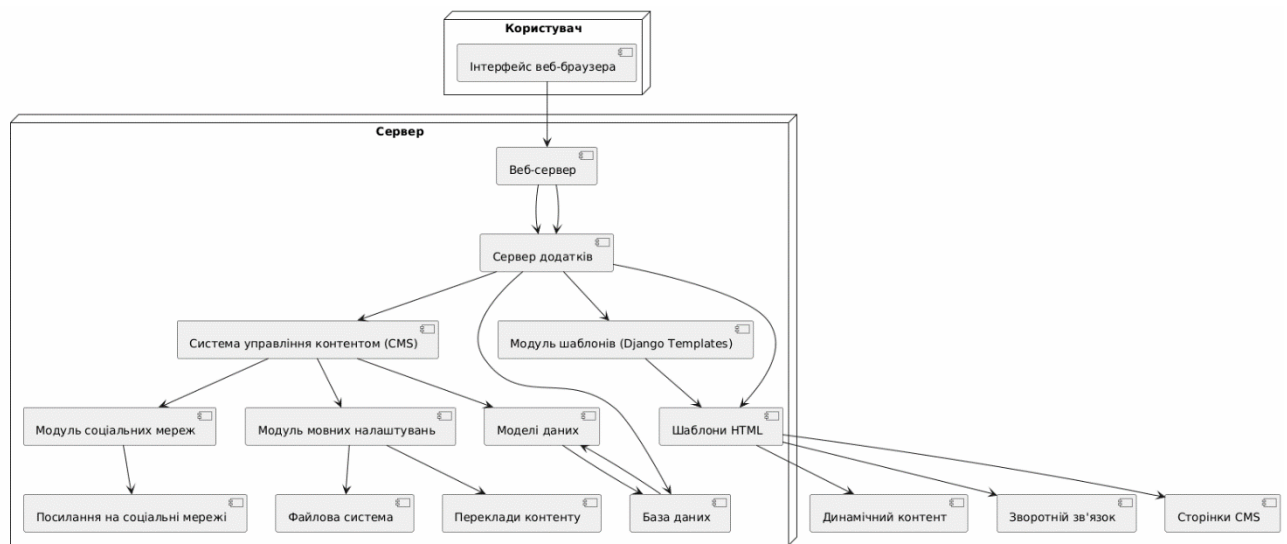


Рисунок 2.11. Діаграма розгортання

Виведення результатів на клієнтську частину: після обробки запиту результат передається клієнтській частині, яка динамічно оновлює контент на сторінці, забезпечуючи комфортну взаємодію з користувачем.

Завдяки модульній архітектурі система легко адаптується до змін. Це дозволяє додавати нові функціональні модулі (наприклад, нові типи контенту або інтеграції з іншими системами) без необхідності змінювати основну частину.

Система повинна бути здатною до розширення в разі росту кількості користувачів або контенту. Це забезпечується за рахунок правильної організації бази даних, кешування та оптимізації запитів.

Враховуючи, що система буде обробляти чутливі дані, забезпечення безпеки є невід'ємною частиною архітектури. Це включає шифрування даних, перевірку доступу та захист від можливих атак.

Архітектура системи управління контентом для малого бізнесу повинна бути спроектована так, щоб забезпечити зручність використання та високий рівень безпеки при зростаючих вимогах до масштабованості та гнучкості. Всі

компоненти системи, включаючи користувачів і їх ролі, серверну та клієнтську частини, базу даних і механізм безпеки, мають бути організовані таким чином, щоб забезпечити ефективну роботу системи, зберігаючи при цьому простоту управління і налаштування. Враховуючи надані діаграми та моделі, архітектура може бути адаптована для потреб малого бізнесу, дозволяючи йому ефективно працювати з контентом при мінімальних ресурсах.

2.3. Інтерфейс користувача

Інтерфейс користувача (UI) є одним із найважливіших елементів розробки будь-якої програмної системи, зокрема системи управління контентом (CMS). У випадку малих бізнесів, які часто не мають власних великих технічних команд, зручність, простота та інтуїтивно зрозумілий дизайн інтерфейсу можуть бути вирішальними факторами для ефективної роботи з CMS. Розробка інтерфейсу користувача для такої системи повинна враховувати специфіку цільової аудиторії, включаючи технічні можливості та рівень підготовки користувачів. Ідеальний інтерфейс дозволяє користувачам виконувати всі необхідні операції з контентом без додаткових труднощів, що особливо важливо для малих підприємств, які шукають ефективність і швидкість у використанні таких інструментів.

Інтерфейс CMS повинен бути гнучким, забезпечуючи персоналізацію для різних категорій користувачів (адміністраторів, редакторів, контент-менеджерів тощо), підтримувати багато мовних інтерфейсів, що також є важливим для малих бізнесів, які мають різноманітні аудиторії або працюють на міжнародних ринках. Важливою вимогою до інтерфейсу є також зручність мобільної версії, оскільки багато користувачів здійснюють доступ до системи через мобільні пристрої.

Проектуючи інтерфейс для CMS, необхідно враховувати основні принципи дизайну, які забезпечують зручність користувачів, ефективність

роботи з системою та позитивний досвід користування. Найважливіші принципи описані в таблиці 2.1.

Таблиця 2.1

Принципи дизайну для забезпечення зручності користувачів,
ефективності роботи з системою та позитивного досвіду користувачів

№	Принцип	Опис
1	Простота та мінімалізм	Усі елементи інтерфейсу повинні бути чітко розташовані, а користувач повинен мати можливість швидко орієнтуватися в системі. Занадто велика кількість елементів, що не мають значення для виконання основних функцій, лише відволікають увагу і створюють складнощі в роботі
2	Інтуїтивно зрозумілий дизайн	Потрібно забезпечити доступ до основних функцій через прості, зрозумілі та природні для користувачів взаємодії. Кнопки, посилання та інші елементи управління повинні бути позначені зрозумілими і значущими ярликами
3	Адаптивність	Система повинна забезпечувати коректне відображення на різних пристроях (ПК, планшетах, смартфонах), що дозволяє користувачам працювати з CMS у будь-який час і в будь-якому місці
4	Зворотний зв'язок	Важливо надавати користувачу зворотний зв'язок під час взаємодії з системою. Наприклад, підтвердження відправки форми, помилки введення даних або повідомлення про успішну операцію
5	Багатофункціональність без перевантаження	Система повинна бути багатофункціональною і забезпечувати можливості для адміністрування контенту, його редагування та публікації, але це не повинно призводити до складності інтерфейсу

Інтерфейс CMS для малого бізнесу, базуючись на наведених принципах, повинен містити кілька основних компонентів, що представлені в таблиці 2.2.

Таблиця 2.2

Компоненти інтерфейсу CMS

№	Компоненти	Опис
1	Головна панель навігації	Це місце, де користувачі можуть отримати доступ до основних функцій CMS, таких як створення нових сторінок, редагування існуючих сторінок, налаштування мовних параметрів та інше. Усі основні дії повинні бути доступні з головного меню без необхідності переходити між численними вкладками чи екранними переходами
2	Мультимедійна підтримка	Для малого бізнесу важливо, щоб система CMS підтримувала роботу з різними типами контенту: текстом, зображеннями, відео, а також забезпечувала їх зручну організацію та публікацію
3	Редактор контенту	Однією з основних частин інтерфейсу є текстовий редактор. Цей редактор повинен бути простим у використанні, з можливістю форматування тексту, додавання медіа-елементів, зміни структури контенту без необхідності вивчати складні коди
4	Управління користувачами та ролями	Для CMS важливо забезпечити функціональність з управління правами доступу, щоб різні категорії користувачів мали можливість редагувати, публікувати або переглядати контент відповідно до своїх ролей у системі
5	Інтерфейс для перегляду і редагування сторінок	Користувач повинен мати можливість переглядати існуючі сторінки, їх контент та структуру, а також здійснювати редагування за допомогою зручного інтерфейсу, де кожен елемент сторінки можна змінити або додати новий
6	Мовні налаштування	Інтерфейс CMS повинен підтримувати можливість зміни мовних налаштувань, що дозволяє підприємствам з різними аудиторіями або у різних регіонах налаштовувати інтерфейс системи на потрібну мову

Головна сторінка CMS для малого бізнесу повинна бути простою і зрозумілою, з чіткою навігацією. Вона повинна містити основні функції, такі як список доступних сторінок, можливість створення нових сторінок та налаштування. Зрозумілий і організований головний екран дає змогу користувачу швидко знайти всі важливі елементи та налаштування.

На головній сторінці також можна додати елементи, що підвищують ефективність роботи, наприклад, відображення останніх змін або попереднього перегляду контенту. Важливою функцією є можливість швидко відкрити та редагувати сторінки, не переходячи між різними розділами CMS.

Редагування сторінок є одним з найбільш важливих елементів CMS, оскільки контент є основною складовою роботи системи. Інтерфейс редагування сторінок повинен бути максимально простим і інтуїтивно зрозумілим, з можливістю вставляти та редагувати текст, додавати зображення, відео, таблиці та інші елементи.

У даному випадку використовується текстовий редактор з візуальним інтерфейсом, що дозволяє редагувати контент без необхідності знання програмування або HTML-коду. Користувач може додавати, змінювати або видаляти елементи безпосередньо з графічного інтерфейсу.

У CMS для малого бізнесу важливу роль відіграють налаштування контенту. Це може включати мета-дані сторінок, їх теги, опис, ключові слова та інші елементи, що допомагають у пошуковій оптимізації та управлінні контентом. Цей інтерфейс повинен бути доступним і зрозумілим, даючи змогу менеджерам зручним чином налаштовувати всі елементи сторінок.

Інтерфейс CMS повинен підтримувати роботу з мультимедійними ресурсами, такими як зображення, відео та аудіо. Мультимедіа може значно покращити сприйняття контенту, тому можливість інтегрувати їх безпосередньо в процес редагування сторінок є важливою частиною інтерфейсу.

Інтерфейс користувача системи управління контентом має критичне значення для ефективності роботи з CMS, особливо для малого бізнесу, де час і ресурси обмежені. Хороший інтерфейс зменшує навчальну криву, дозволяючи користувачам швидко освоїти систему та ефективно використовувати її функції. Простота, зручність та адаптивність – основні принципи, які повинні бути враховані при проектуванні інтерфейсу CMS. В кінцевому рахунку, добре

спроектований інтерфейс забезпечує не тільки задоволення користувачів, але й підвищує продуктивність і ефективність роботи системи в цілому.

Висновки до розділу 2

Проектування системи управління контентом (CMS) є важливим етапом у розробці інструментів для малих підприємств, що потребують зручного та ефективного способу керувати своїм цифровим контентом. Це завдання вимагає ретельного підходу до кожного елементу системи, щоб забезпечити не лише її функціональність, але й простоту у використанні, особливо для користувачів без глибоких технічних знань. У рамках розробки CMS для малого бізнесу необхідно враховувати численні вимоги, починаючи від функціональних можливостей і завершуючи детальним дизайном інтерфейсу, що має бути інтуїтивно зрозумілим і зручним у повсякденному використанні.

Одним із найважливіших елементів, що визначають ефективність CMS, є її функціональні можливості. Вони повинні відповідати сучасним вимогам бізнесу, враховуючи швидкий розвиток технологій та постійно зростаючі вимоги до роботи з великими обсягами даних. Важливою складовою є наявність інструментів для SEO-оптимізації, що дозволяють підприємствам ефективно взаємодіяти з пошуковими системами, залучати нових клієнтів і підвищувати видимість своїх онлайн-ресурсів. Підтримка багатомовності, у свою чергу, стає необхідною функцією для малих бізнесів, що працюють на різних ринках або мають клієнтів з різних куточків світу. Це забезпечує не тільки комфорт для кінцевих користувачів, але й підвищує гнучкість системи в умовах різних культурних та мовних середовищ.

Особливу увагу слід приділити безпеці даних, адже малий бізнес часто оперує чутливою інформацією про клієнтів, контракти та фінансові операції. Це вимагає наявності надійних механізмів захисту від несанкціонованого доступу, таких як шифрування даних і перевірка прав доступу для різних категорій користувачів. Забезпечення високого рівня безпеки є критично важливим для успішного функціонування системи і довіри з боку користувачів.

Не менш важливим є питання архітектури системи. Вона має бути спроектована таким чином, щоб забезпечити масштабованість, гнучкість і ефективність роботи з контентом навіть при обмежених ресурсах. Архітектура CMS повинна дозволяти швидко адаптувати систему до нових вимог бізнесу, легко змінюючи або розширюючи функціональність без шкоди для її стабільності. Вона повинна включати такі компоненти, як бази даних, механізми захисту і управління користувачами, а також зручний інтерфейс для адміністрування контенту. Крім того, система повинна бути здатною підтримувати великі обсяги інформації та забезпечувати високу швидкість обробки запитів.

Однак важливим елементом, що впливає на успішність впровадження CMS, є її інтерфейс. Для малого бізнесу це часто є найбільш критичним моментом, оскільки менеджери та співробітники можуть не мати глибоких технічних знань. Інтерфейс користувача CMS повинен бути максимально простим, зручним і адаптивним. Принцип мінімалізму є ключовим для створення інтерфейсу, який би забезпечував безперешкодну взаємодію з системою. Всі елементи управління повинні бути на своїх місцях і надавати користувачеві можливість швидко знайти необхідні інструменти для роботи з контентом. Кожен компонент інтерфейсу повинен бути інтуїтивно зрозумілим, що дозволить користувачам, навіть тим, хто має обмежений досвід роботи з подібними системами, без труднощів освоїти всі необхідні функції.

Важливо також забезпечити адаптивність інтерфейсу, оскільки в умовах сьогоденішньої динамічної бізнес-середовища мобільний доступ до системи стає невід'ємною частиною її роботи. Підприємства повинні мати можливість працювати з CMS не лише з настільних комп'ютерів, а й з мобільних пристроїв, оскільки це дозволяє менеджерам та співробітникам залишатися на зв'язку з системою в будь-який час і в будь-якому місці. Ця гнучкість особливо важлива для малих бізнесів, де обмеженість ресурсів може вимагати максимальної мобільності.

Проектування інтерфейсу користувача також передбачає інтеграцію з іншими системами, які використовуються підприємством, такими як інструменти аналітики, CRM-системи або системи управління товарами та замовленнями. Взаємодія між цими системами може значно спростити робочі процеси і підвищити ефективність роботи підприємства, дозволяючи уникати дублювання даних і забезпечувати єдину точку доступу до всієї необхідної інформації.

Отже, проектування системи управління контентом для малого бізнесу передбачає розгляд всіх етапів розробки, від функціональних вимог і архітектури до створення зручного та адаптивного інтерфейсу. Врахування усіх елементів, пов'язаних із безпекою, гнучкістю, ефективністю роботи з контентом і зручністю користувачів, є необхідною умовою для успішної реалізації такої системи. Тільки комплексний підхід до розробки CMS дозволить створити інструмент, що буде максимально ефективним, простим у використанні і відповідатиме вимогам сучасного малого бізнесу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОЕКТУ

3.1. Розробка основних модулів CMS

Система управління контентом (CMS) є важливою складовою для ефективної роботи будь-якого веб-сайту, особливо для малого бізнесу, де доступність, простота у використанні та швидкість розгортання мають велике значення. CMS дозволяє користувачам, зокрема адміністратору та контент-менеджеру, легко додавати, редагувати та організовувати контент на сайті без необхідності глибоких технічних знань. Реалізація такої системи потребує розробки основних модулів (рисунок 3.1), що забезпечують функціональність сайту, інтерфейс для користувачів, а також безпеку та ефективність роботи.

Розробка модулів CMS для малого бізнесу вимагає визначення чітких функціональних блоків, які мають бути реалізовані для забезпечення зручності в управлінні контентом, оптимізації його поширення та адаптації під вимоги конкретного бізнесу. Важливим етапом є розгляд процесу розробки основних модулів системи, їх інтеграція в єдину платформу та забезпечення тестування їх функціональності для досягнення високої якості роботи системи.

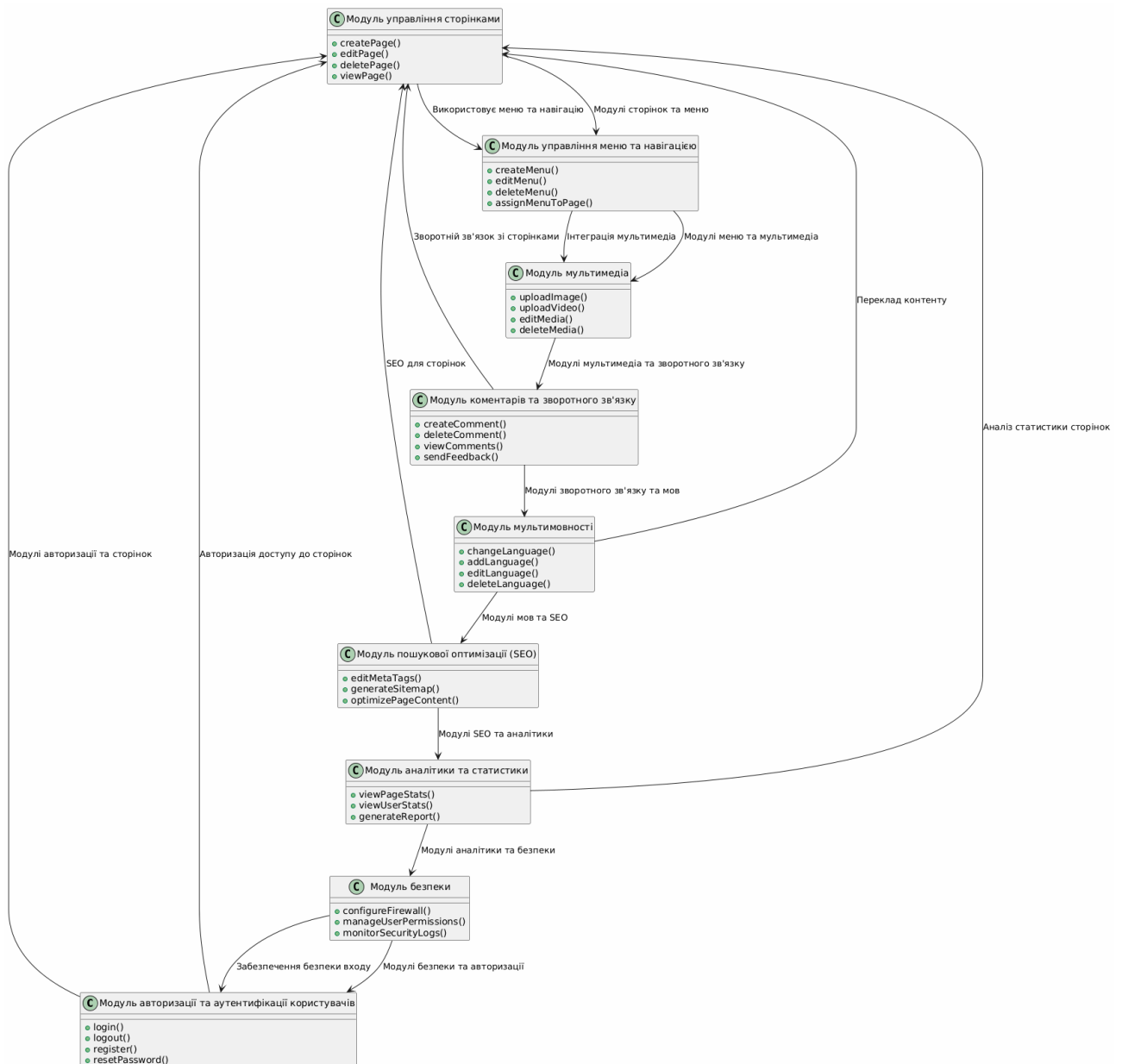


Рисунок 3.1. Основні модулі системи

3.1.1. Модуль авторизації та аутентифікації користувачів

Безпека та контроль доступу є першочерговими вимогами для будь-якої CMS. Модуль авторизації та аутентифікації користувачів служить основою для захисту ресурсу від несанкціонованого доступу. Реалізація цього компонента включає в себе два основні етапи: реєстрація користувачів і перевірка їх прав на доступ до різних частин веб-сайту.

Для забезпечення аутентифікації користувачів використовуються логіни та паролі, що зберігаються в зашифрованому вигляді в базі даних. Це дає змогу

забезпечити високу безпеку особистих даних користувачів, оскільки навіть у разі витоку даних доступ до акаунтів буде неможливий без пароля. Крім того, можна впровадити функцію двоетапної верифікації, яка значно підвищує рівень захисту, зокрема шляхом використання тимчасових кодів, що надсилаються на мобільні телефони користувачів.

Наявність можливості відновлення пароля через електронну пошту також є важливим компонентом, який забезпечує користувачам можливість відновити доступ до своїх акаунтів у разі втрати пароля.

3.1.2. Модуль управління сторінками

Основною метою CMS є ефективне управління контентом, і модуль управління сторінками є ключовим елементом, що реалізує цю задачу. Цей модуль дозволяє створювати, редагувати та публікувати різні сторінки сайту, зберігаючи структуру та організацію контенту. При цьому важливим є надання можливості керування мета-даними кожної сторінки, такими як мета-опис, ключові слова і заголовки, що дозволяє значно покращити видимість сторінок у пошукових системах (SEO).

Кожна сторінка має свій унікальний ідентифікатор і може містити різні види контенту: текстові матеріали, зображення, відео, таблиці тощо. Важливим є забезпечення функціональності попереднього перегляду сторінки перед її публікацією, що дозволяє адміністратору перевірити вигляд і зміст перед публікацією на сайті. Інтерфейс для створення і редагування сторінок повинен бути інтуїтивно зрозумілим, що дозволяє навіть непідготовленим користувачам швидко освоїти роботу з системою.

Модуль управління сторінками також включає функціональність для роботи з шаблонами та їх інтеграцією з конкретними елементами сторінки, забезпечуючи її привабливий вигляд.

3.1.3. Модуль управління меню та навігацією

Кожен веб-сайт, незалежно від його складності, потребує чіткої і зручної навігації. Модуль управління меню та навігацією дозволяє створювати й редагувати структуру меню, яка є основним інструментом навігації для користувачів. У даному модулі реалізується механізм додавання нових пунктів меню, а також налаштування їхнього порядку. Адміністратор може налаштувати як основне меню сайту, так і створювати підменю для окремих розділів, що дає змогу організувати структуру сайту відповідно до його специфіки.

Модуль також передбачає функціональність для зміни видимості пунктів меню, залежно від прав доступу користувача, а також можливість приховувати пункти, які наразі не є актуальними чи доступними. Це дає змогу гнучко керувати навігацією і робити сайт зручним для різних груп користувачів.

3.1.4. Модуль мультимедіа

Мультимедіа є важливим компонентом будь-якого сучасного веб-сайту, і модуль мультимедіа відповідає за зберігання, завантаження та управління медіафайлами. Веб-сайти для малого бізнесу часто використовують зображення, відео та інші медіа-ресурси для підвищення привабливості своїх сторінок і забезпечення інтерактивності з відвідувачами.

Модуль мультимедіа дозволяє зручно завантажувати файли, додавати їх до відповідних сторінок і управляти ними через інтерфейс адміністратора. Важливою частиною функціональності є можливість редагувати медіафайли, змінювати їх розміри, додавати водяні знаки або оптимізувати їх для швидшого завантаження. Крім того, зображення та відео можуть бути представлені в галереях, що дозволяє зручно переглядати велику кількість медіа-ресурсів без перевантаження інтерфейсу.

3.1.5. Модуль коментарів та зворотного зв'язку

Для залучення користувачів до взаємодії з контентом необхідно впровадити систему коментарів, що дозволяє залишати відгуки під статтями, товарами чи іншими елементами контенту. Модуль коментарів передбачає можливість надавати оцінки, коментувати окремі публікації, а також здійснювати модерацію коментарів, щоб уникнути спаму або неетичних висловлювань.

Крім того, важливим є модуль зворотного зв'язку, що дає змогу користувачам надсилати запити або залишати повідомлення для адміністратора. Це може бути корисно для бізнесу, оскільки дозволяє отримувати інформацію безпосередньо від клієнтів, що покращує взаємодію і сприяє розвитку.

3.1.6. Модуль мультимовності

Інтернаціоналізація є важливою складовою для CMS, орієнтованої на глобальну аудиторію. Модуль мультимовності дає можливість створювати та редагувати контент на кількох мовах, що особливо важливо для малих бізнесів, що прагнуть розширити свою присутність на міжнародних ринках.

Модуль мультимовності може передбачати інтеграцію з перекладацькими сервісами або можливість ручного введення перекладів для кожної сторінки сайту. Завдяки цьому користувачі з різних регіонів зможуть отримати доступ до інформації на своєму рідному мовному середовищі, що значно підвищує зручність використання ресурсу.

3.1.7. Модуль пошукової оптимізації (SEO)

Пошукова оптимізація є критичним елементом для успіху будь-якого сайту в Інтернеті. Модуль SEO відповідає за налаштування всіх елементів, пов'язаних з підвищенням видимості веб-сайту в пошукових системах. Це включає в себе налаштування мета-даних для кожної сторінки (заголовки, опис,

ключові слова), використання структурованих даних, а також оптимізацію швидкості завантаження сторінок.

Модуль також може включати автоматичні рекомендації щодо покращення SEO на основі аналізу контенту сайту. Це дає змогу підтримувати сайт на високому рівні з точки зору пошукової оптимізації без необхідності постійно вивчати нові тенденції в цій галузі.

3.1.8. Модуль аналітики та статистики

Модуль аналітики дозволяє власникам сайту отримувати необхідні дані про відвідуваність ресурсу, популярні сторінки, поведінку користувачів на сайті та інші важливі метрики. Це дозволяє ефективно управляти контентом, відстежувати його ефективність і адаптувати стратегії розвитку відповідно до отриманих даних.

Розробка такого модуля включає в себе інтеграцію з популярними аналітичними сервісами (наприклад, Google Analytics), а також створення власних інструментів для збору статистичних даних.

3.1.9. Модуль безпеки

Безпека веб-сайту є одним з найважливіших елементів для забезпечення стабільної та безпечної роботи. Модуль безпеки покликаний захищати сайт від різноманітних атак, таких як SQL ін'єкції, XSS (міжсайтові сценарії) та інші уразливості.

Для забезпечення безпеки важливо використовувати методи хешування паролів, шифрування даних, а також налаштування багатофакторної аутентифікації для адміністративних доступів.

Розробка основних модулів для системи управління контентом для малого бізнесу є складним і багатогранним процесом, що вимагає уваги до деталей і гнучкості в підході. Реалізація таких модулів як авторизація, управління сторінками, мультимедіа та SEO дозволяє створити потужну платформу для адміністрування веб-сайту. Ретельне тестування кожного з

модулів гарантує їх правильну інтеграцію в загальну систему і забезпечує функціональність та безпеку веб-сайту для користувачів.

Кожен з цих модулів сприяє досягненню основних цілей CMS — зручності в управлінні контентом, забезпеченні інтерактивності з користувачами та ефективному моніторингу ефективності роботи сайту.

3.2. Тестування системи

Тестування є одним із найважливіших етапів у розробці будь-якої інформаційної системи, оскільки забезпечує її коректну роботу, відповідність вимогам та зручність використання кінцевими користувачами. Особливо важливо ретельно перевіряти систему управління контентом (CMS), оскільки вона є основним інструментом для адміністрування веб-сайтів, зберігання і обробки контенту та взаємодії з користувачами. Якщо система містить помилки чи недоліки, це може призвести до серйозних проблем у роботі веб-ресурсу та знижує його ефективність.

Тестування системи управління контентом для малого бізнесу має на меті забезпечити правильну взаємодію між різними компонентами, перевірити функціональність усіх основних можливостей, а також підтвердити, що система зручна та ефективна для користувачів. Враховуючи різноманітність функцій CMS — від створення сторінок до обробки форм зворотного зв'язку — важливо виконати тестування кожного елементу окремо і в комплексі, щоб уникнути помилок і переконатися в стабільності роботи системи.

3.2.1. Покрокове виконання проєкту

На самому початку роботи системи управління контентом (CMS) для малого бізнесу необхідно налаштувати середовище для виконання програми. Веб-додаток базується на Django, популярному фреймворку для розробки веб-сайтів на Python. Тому, щоб запустити програму, потрібно налаштувати Django-проєкт та забезпечити базову конфігурацію.

Налаштування середовища: Першим етапом є створення віртуального середовища, де будуть розташовані всі залежності програми. За допомогою Python і `pip` (менеджера пакетів Python) користувачеві потрібно створити та активувати віртуальне середовище.

```
python3 -m venv myenv
source myenv/bin/activate # для Linux/macOS
myenv\Scripts\activate   # для Windows
```

Встановлення необхідних бібліотек: Після активації середовища необхідно встановити всі необхідні бібліотеки та модулі для Django. Це включає сам Django, а також інші бібліотеки для роботи з базами даних, наприклад, PostgreSQL або MySQL, а також додаткові бібліотеки для роботи з медіафайлами чи кешуванням.

```
pip install django
pip install psycopg2 # для PostgreSQL
pip install Pillow   # для роботи з медіа
```

Створення проекту Django: Після налаштування середовища, можна створити новий проект Django. Це дасть основу для подальшої розробки, включаючи створення базових файлів конфігурації.

```
django-admin startproject cms_project
```

Конфігурація бази даних: Важливим етапом є налаштування підключення до бази даних. За допомогою Django можна легко працювати з різними типами баз даних, у тому числі з реляційними базами, такими як PostgreSQL або MySQL.

Міграція бази даних (рисунок 3.2): Після налаштування підключення потрібно створити базу даних та здійснити міграцію для того, щоб всі таблиці, які потрібні для роботи додатка, були створені.

python manage.py migrate

```
PS D:\work_studlab\CMS\my_cms> python manage.py migrate
>>
System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions, sites
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying main.0001_initial... OK
  Applying main.0002_menu_sociallink_remove_product_category_and_more... OK
  Applying main.0003_page_featured_image_alter_sociallink_icon_pageview_and_more... OK
  Applying main.0004_delete_role... OK
  Applying main.0005_alter_pageview_referrer... OK
  Applying sessions.0001_initial... OK
  Applying sites.0001_initial... OK
  Applying sites.0002_alter_domain_unique... OK
```

Рисунок 3.2. Міграція бази даних

Старт серверу (рисунок 3.3): Після налаштування та міграції бази даних, можна запустити сервер Django і перевірити, чи працює додаток.

python manage.py runserver

При цьому веб-додаток буде доступний за адресою `http://127.0.0.1:8000`.

```
PS D:\work_studlab\CMS\my_cms> python manage.py runserver 5053
INFO 2025-01-20 11:48:48,650 autoreload Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)

System check identified 1 issue (0 silenced).
January 20, 2025 - 11:48:49
Django version 5.1.5, using settings 'settings'
Starting development server at http://127.0.0.1:5053/
Quit the server with CTRL-BREAK.
```

Рисунок 3.3. Старт серверу

На цьому етапі запуск програми завершено, і користувач може перейти з головної сторінки (рисунок 3.4) до створення сторінок та налаштування контенту через адміністративну панель.

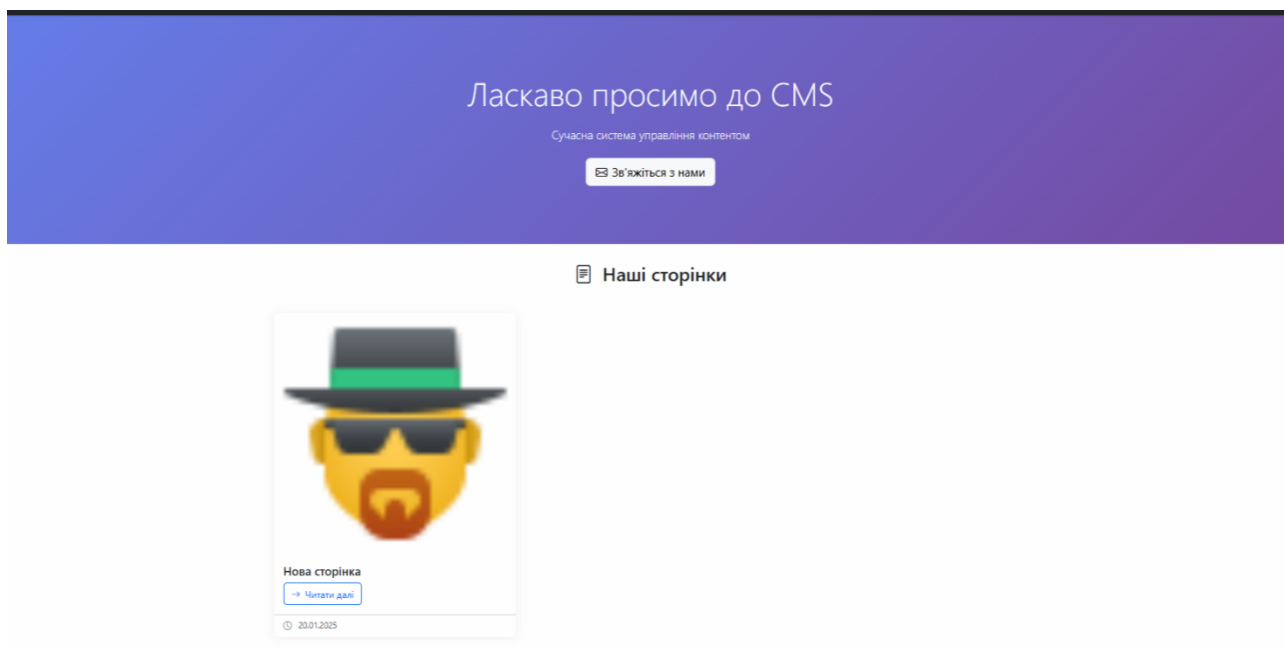


Рисунок 3.4. Головна сторінка

Крок 2: Створення основних шаблонів

Другий етап роботи програми пов'язаний із налаштуванням основних шаблонів, які використовуватимуться для рендерингу сторінок. У CMS системах шаблони грають важливу роль, оскільки вони визначають, як контент буде відображатися користувачам на різних сторінках сайту.

Створення базового шаблону: Уся структура сайту ґрунтується на базовому шаблоні, який містить загальні елементи для всіх сторінок: шапку, меню, футер. Це дозволяє зберегти консистентність вигляду сайту. У реалізованому коді це реалізовано в шаблоні `base.html`.

Блоки контенту: За допомогою тегів `{% block %}`, як у `base.html`, визначаються місця для вставки специфічного контенту на кожній окремій сторінці. Шаблон може містити певний статичний контент, наприклад, меню, яке не змінюється, а також місце для динамічного виведення, яке буде наповнюватися на кожній окремій сторінці.

Підключення стилів та скриптів: Важливим елементом є підключення зовнішніх стилів і скриптів, таких як Bootstrap, який використовується для

оформлення сторінок. Цей фреймворк дозволяє створювати адаптивні і зручні інтерфейси користувача з мінімальними зусиллями.

Інтернаціоналізація: Важливою частиною системи є підтримка різних мов. Для цього використовуються теги `{% trans %}`, які дозволяють здійснювати переклад елементів на різні мови. Це важливо для CMS систем, які повинні бути доступні для широкої аудиторії.

Після налаштування основних шаблонів система готова до створення конкретних сторінок контенту.

Крок 3: Створення сторінок контенту

Після налаштування базових шаблонів наступним кроком є створення контенту для сторінок сайту. У системах управління контентом (CMS) кожна сторінка представляє собою певний блок інформації, який може бути динамічно змінений адміністратором чи редактором сайту. Задля зручності й гнучкості, CMS-система дозволяє створювати різні типи сторінок, включаючи статичні та динамічні.

Створення моделі для сторінок: В першу чергу, необхідно створити модель для сторінок, яка включає основні поля, що описують сторінку: заголовок, вміст, мета-опис, ключові слова, дата останнього оновлення, зображення тощо. Модель для сторінок у Django виглядатиме наступним чином:

```
from django.db import models
from django.utils import timezone

class Page(models.Model):
    title = models.CharField(max_length=255)
    content = models.TextField()
    meta_title = models.CharField(max_length=255,
blank=True, null=True)
    meta_description = models.TextField(blank=True,
null=True)
    meta_keywords = models.CharField(max_length=255,
blank=True, null=True)
```

```

    featured_image =
models.ImageField(upload_to='featured_images/',
blank=True, null=True)
    created = models.DateTimeField(auto_now_add=True)
    modified = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.title
    def get_absolute_url(self):
        return f'/page/{self.id}/'

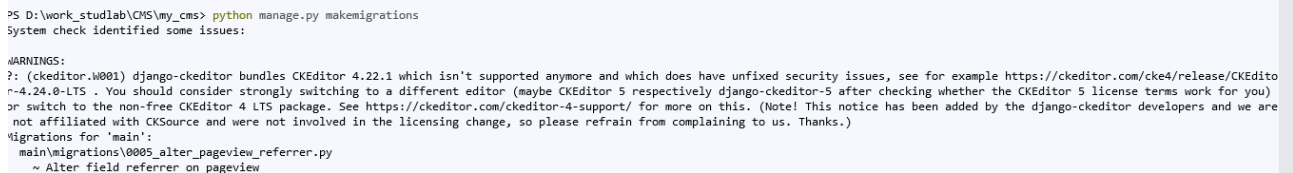
```

Міграція моделей (рисунок 3.5): Після того як модель створена, потрібно виконати міграцію, щоб ці зміни відобразилися в базі даних:

```

python manage.py makemigrations
python manage.py migrate

```



```

PS D:\work_studlab\CMS\my_cms> python manage.py makemigrations
System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)

Migrations for 'main':
  main/migrations/0005_alter_pageview_referrer.py
    ~ Alter field referrer on pageview

```

Рисунок 3.5. Міграція моделей

Створення адміністративної панелі: Оскільки сторінки можуть бути створені або редаговані адміністраторами, необхідно налаштувати адміністративну панель Django для зручного управління контентом (рисунок 3.6, рисунок 3.7). Для цього потрібно зареєструвати модель сторінки в адмінці:

```

from django.contrib import admin
from .models import Page

@admin.register(Page)
class PageAdmin(admin.ModelAdmin):
    list_display = ['title', 'created', 'modified']
    search_fields = ['title', 'content']

```

Django адміністрування

Ім'я користувача:

Пароль:

Увійти

Рисунок 3.6. Вхід до адміністративної панелі

Django адміністрування

Адміністрування сайту

MAIN		
Меню	+ Додати	Змінити
Перегляди сторінок		Змінити
Повідомлення	+ Додати	Змінити
Соціальні мережі	+ Додати	Змінити
Сторінки	+ Додати	Змінити

АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ		
Групи	+ Додати	Змінити
Користувачі	+ Додати	Змінити

САЙТИ		
Сайти	+ Додати	Змінити

Недавні дії

Мої дії

Немає

Рисунок 3.7. Панель адміністрування

Створення елементів через адміністративну панель: Після реєстрації моделі в адміністративній панелі, адміністратори можуть створювати нові сторінки(рисунок 3.8), додавати меню (рисунок 3.9), соціальні мережі (рисунок 3.10), повідомлення (рисунок 3.11) та сайту (рисунок 3.12) через веб-інтерфейс, вводячи заголовок, вміст та інші необхідні поля.

Рисунок 3.8. Створення нової сторінки

Рисунок 3.9. Створення меню

Django адміністрування ВІТАЄМО, **ADMIN** [ДИВІТИСЯ САЙТ](#) / [ЗМІНИТИ ПАРОЛЬ](#) / [ВИЙТИ](#)

Домівка > Main > Соціальні мережі > Додати Соціальна мережа

Почніть писати для фільтру...

MAIN

Меню [+ Додати](#)

Повідомлення [+ Додати](#)

Соціальні мережі [+ Додати](#)

Сторінки [+ Додати](#)

АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ

Групи [+ Додати](#)

Користувачі [+ Додати](#)

« **САЙТИ**

Сайти [+ Додати](#)

Додати Соціальна мережа

Назва:

URL:

Іконка: [Виберіть файл](#) | Файл не вибран

Порядок:

[ЗБЕРЕГТИ](#) [Зберегти і додати інше](#) [Зберегти і продовжити редагування](#)

Рисунок 3.10. Додавання соціальної мережі

Django адміністрування ВІТАЄМО, **ADMIN** [ДИВІТИСЯ САЙТ](#) / [ЗМІНИТИ ПАРОЛЬ](#) / [ВИЙТИ](#)

Домівка > Main > Повідомлення > Додати Повідомлення

Почніть писати для фільтру...

MAIN

Меню [+ Додати](#)

Перегляди сторінок

Повідомлення [+ Додати](#)

Соціальні мережі [+ Додати](#)

Сторінки [+ Додати](#)

АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ

Групи [+ Додати](#)

Користувачі [+ Додати](#)

« **САЙТИ**

Сайти [+ Додати](#)

Додати Повідомлення

Ім'я:

Email:

Повідомлення:

☐ Оброблено

Створено: -

[ЗБЕРЕГТИ](#) [Зберегти і додати інше](#) [Зберегти і продовжити редагування](#)

Рисунок 3.11. Додавання повідомлення

Django адміністрування

Домівка > Сайти > Сайти > Додати сайт

Почніть писати для фільтру...

MAIN

Меню [+ Додати](#)

Перегляди сторінок

Повідомлення [+ Додати](#)

Соціальні мережі [+ Додати](#)

Сторінки [+ Додати](#)

АУТЕНТИФІКАЦІЯ ТА АВТОРИЗАЦІЯ

Групи [+ Додати](#)

Користувачі [+ Додати](#)

САЙТИ

Сайти [+ Додати](#)

Додати сайт

Доменне ім'я:

Відображуване ім'я:

[ЗБЕРЕГТИ](#) [Зберегти і додати інше](#) [Зберегти і продовжити редагування](#)

Рисунок 3.12. Додавання сайту

Крок 4: Рендеринг сторінок через шаблони

Щоб кожна сторінка мала свій інтерфейс на веб-сайті, потрібно створити відповідні шаблони для відображення контенту.

Рендеринг статичних та динамічних сторінок: Для того щоб створити сторінку з динамічним контентом, спочатку необхідно визначити URL-шаблони. У Django це робиться за допомогою системи маршрутизації.

Для цього необхідно створити файл `urls.py` у додатку сторінок, в якому буде описано, як саме звертатися до конкретних сторінок через URL:

```
from django.urls import path
from . import views

urlpatterns = [
    path('<int:id>
    /', views.page_detail, name='page_detail'),
]
```

Контролер для сторінки: Далі створюється відповідний контролер (view), який буде обробляти запит на сторінку. Він викликає модель сторінки та передає її контент до шаблону для рендерингу:

```
from django.shortcuts import render, get_object_or_404
from .models import Page

def page_detail(request, id):
    page = get_object_or_404(Page, id=id)
    return render(request, 'page_detail.html', {'page':
    page})
```

Шаблон для сторінки (рисунок 3.13): Для того щоб відобразити сторінку, створюється шаблон `page_detail.html`, в якому використовується переданий контент:

```
{% extends "base.html" %}

{% block title %}{{ page.title }} - CMS{% endblock %}
```

```
{% block content %}
<div class="container">
  <h1>{{ page.title }}</h1>
  <div>{{ page.content|safe }}</div>
  <p>Оновлено: {{ page.modified|date:"d.m.Y" }}</p>
</div>
{% endblock %}
```

Цей шаблон виводить заголовок сторінки, її вміст та дату останнього оновлення.

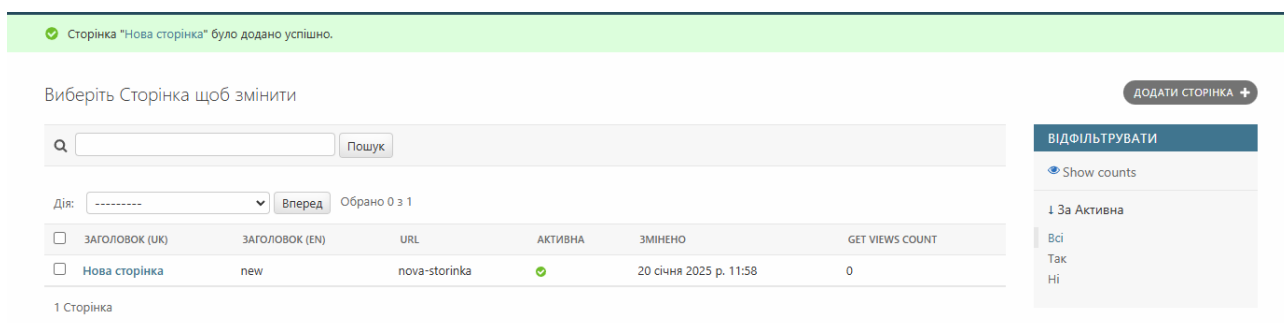


Рисунок 3.13. Дані створених сторінок

Аналогічні дані створені і для інших елементів системи (рисунок 3.14, рисунок 3.15, рисунок 3.16)

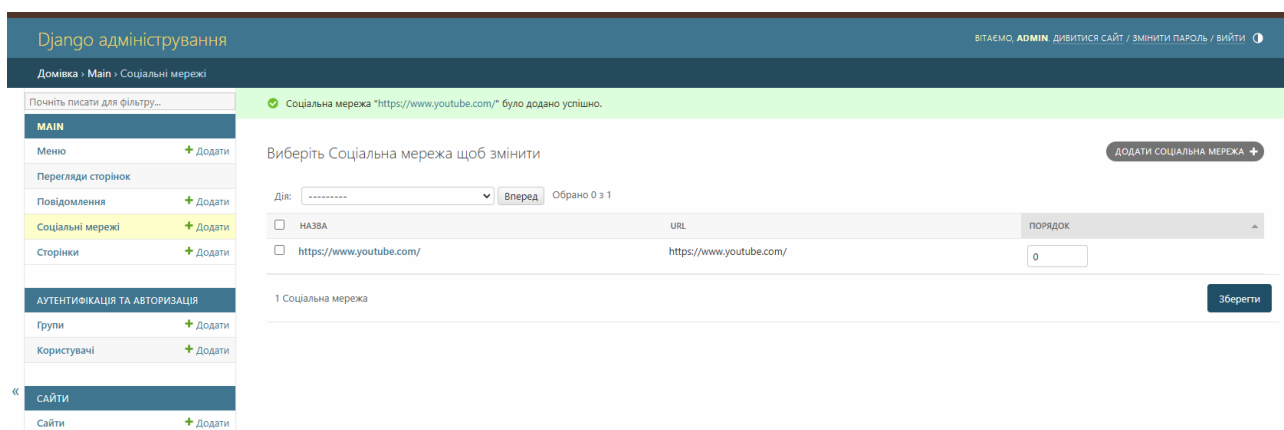


Рисунок 3.14. Дані доданих соціальних мереж

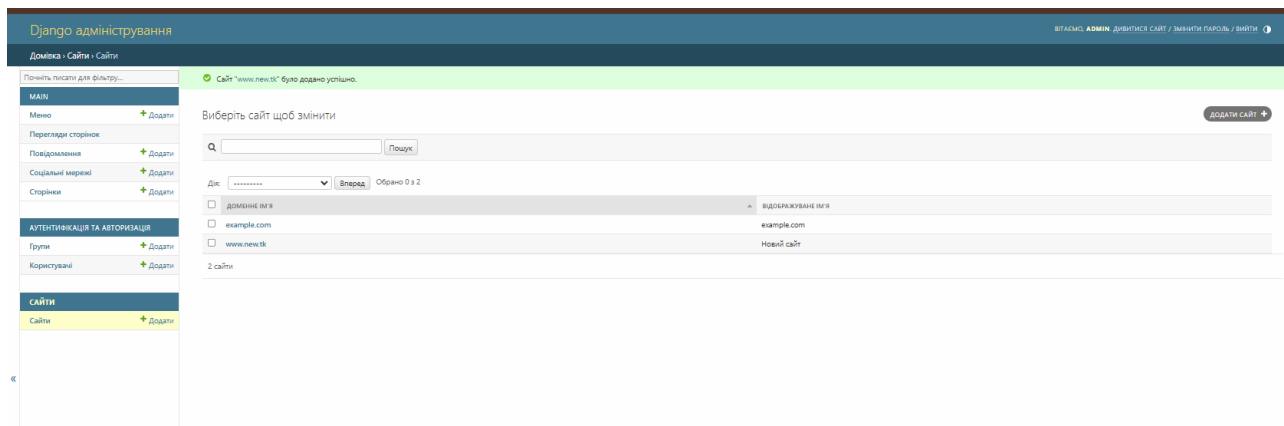


Рисунок 3.15. Дані доданих сайтів

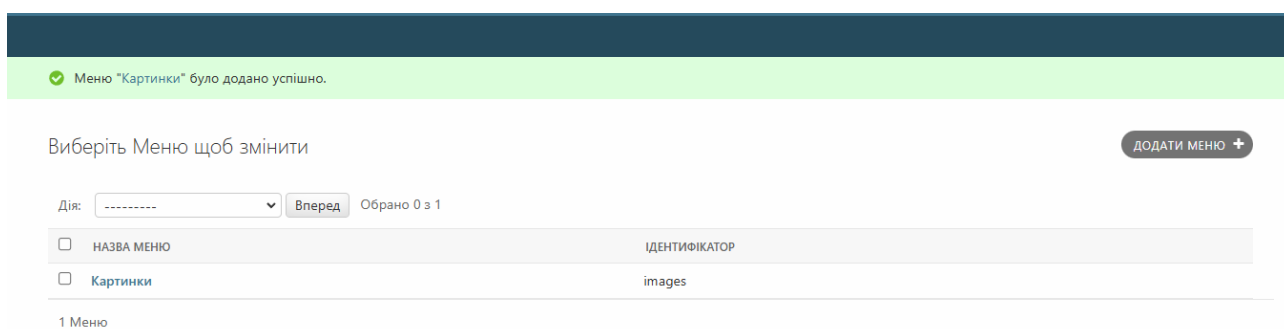


Рисунок 3.16. Дані меню

Крок 5: Обробка форм зворотного зв'язку

У CMS для малого бізнесу важливою функціональністю є можливість отримання зворотного зв'язку від користувачів. Для цього реалізується форма зворотного зв'язку (рисунок 3.17), яка дозволяє користувачам надсилати свої повідомлення або запити адміністраторам.

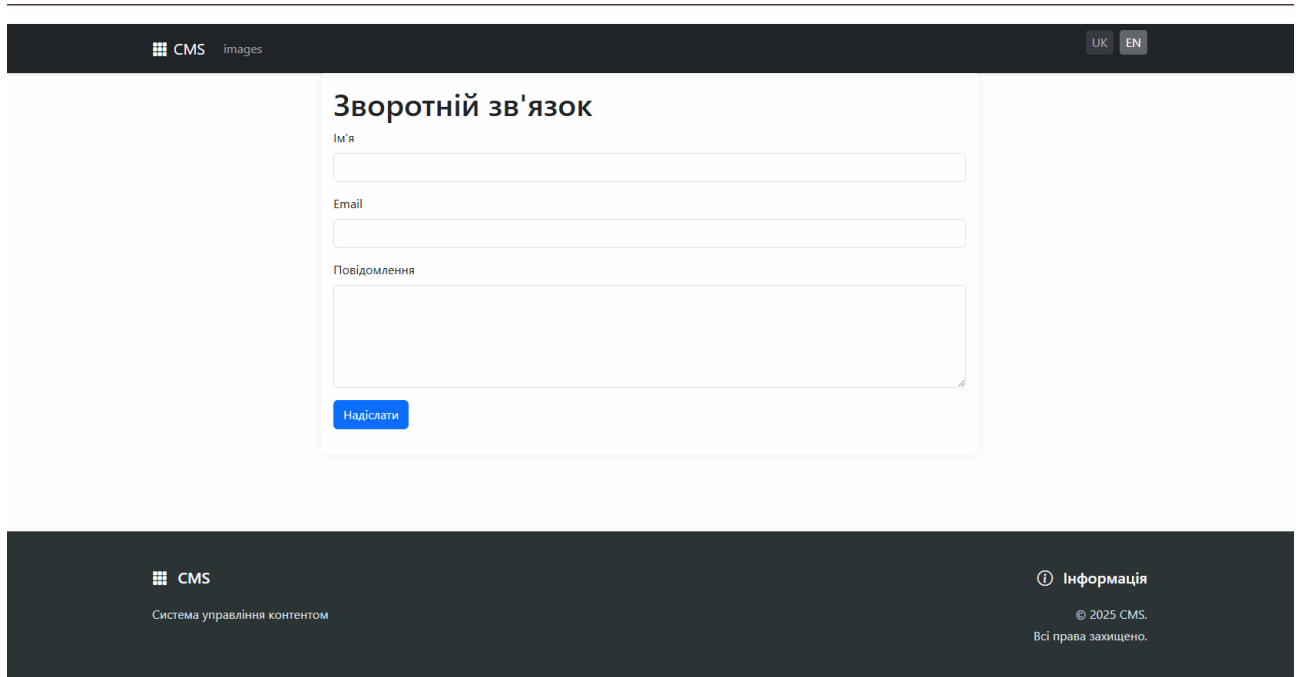


Рисунок 3.17. Форма зворотного зв'язку

Модель для зворотного зв'язку: Спочатку створюється модель форми, яка описує поля, що будуть використані для отримання інформації від користувача:

```
from django import forms

class ContactForm(forms.Form):
    name = forms.CharField(max_length=100, required=True)
    email = forms.EmailField(required=True)
    message = forms.CharField(widget=forms.Textarea,
                              required=True)
```

Шаблон форми зворотного зв'язку: Для того щоб вивести цю форму на сторінці, створюється відповідний шаблон. У шаблоні `feedback.html` будуть виводитися всі поля форми:

```
{% extends "base.html" %}

{% block title %}Зворотній зв'язок - CMS{% endblock %}

{% block content %}
<div class="container">
  <h1>Зворотній зв'язок</h1>
  <form method="post">
```

```

        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="btn btn-
primary">Надіслати</button>
    </form>
</div>
{% endblock %}

```

Обробка форми на сервері: У контролері feedback обробляється отримана форма та, якщо форма заповнена правильно, надається підтвердження про успішне надсилання:

```

from django.shortcuts import render
from .forms import ContactForm

def feedback(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            # Логіка для обробки форми (наприклад,
            збереження в базі даних або відправка на email)
            return render(request, 'message_sent.html')
        else:
            form = ContactForm()
            return render(request, 'feedback.html', {'form':
form})

```

Крок 6: Надсилання підтвердження про успіх

Після того, як повідомлення зворотного зв'язку було надіслано, користувач має отримати підтвердження про успіх. Для цього створюється окремий шаблон, в якому виводиться повідомлення про успішне надсилання.

```

{% extends "base.html" %}

{% block title %}Повідомлення надіслано - CMS{% endblock %}

{% block content %}
<div class="alert alert-success">
    <h4>Дякуємо!</h4>

```

```

    <p>Ваше повідомлення успішно надіслано. Ми зв'яжемося
з вами найближчим часом.</p>
    <a href="{% url 'home' %}" class="btn btn-primary">На
головну</a>
</div>
{% endblock %}

```

Крок 7: Завершення роботи програми

Після завершення всіх основних функцій веб-сайту система готова до запуску в продукційному середовищі. Етапи, які необхідно виконати на завершення розробки:

Перевірка і тестування: Перш ніж вивести сайт на реальну аудиторію, необхідно провести повне тестування всіх функцій, включаючи перевірку форм, рендерингу сторінок, відображення медіафайлів, а також перевірку продуктивності.

Налаштування безпеки: Важливо налаштувати захист від SQL-ін'єкцій, XSS-атак, а також налаштувати безпечні куки і захист від CSRF.

Оптимізація: Після завершення розробки веб-сайт необхідно оптимізувати для кращої продуктивності. Це включає кешування, мініфікацію CSS і JavaScript, а також налаштування CDN для статичних файлів.

Запуск в продукційному середовищі: Після тестування та налаштування безпеки проект можна запускати на сервері, забезпечивши доступ до нього з інтернету.

3.2.2. Тестування запуску системи та перевірка головної сторінки

Тестування системи починається з її запуску на локальному сервері або хостинговій платформі (рисунок 3.18). Для цього використовують команду `python manage.py runserver`, яка активує веб-сервер Django (рисунок 3.19). Після цього користувач може перейти на головну сторінку за URL-адресою, виведеною у консолі. Важливими етапами є перевірка того, чи коректно завантажується основна сторінка, чи відображаються елементи інтерфейсу, а також чи працюють функціональні кнопки та посилання.

```

PS D:\work_studlab\CMS\my_cms> python manage.py makemigrations
System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)
Migrations for 'main':
  main/migrations/0005_alter_pageview_referrer.py
    ~ Alter field referrer on pageview

```

Рисунок 3.18. Запуск на локальному сервері

```

PS D:\work_studlab\CMS\my_cms> python manage.py migrate
>>
System check identified some issues:

WARNINGS:
?: (ckeditor.W001) django-ckeditor bundles CKEditor 4.22.1 which isn't supported anymore and which does have unfixed security issues, see for example https://ckeditor.com/cke4/release/CKEditor-4.24.0-LTS . You should consider strongly switching to a different editor (maybe CKEditor 5 respectively django-ckeditor-5 after checking whether the CKEditor 5 license terms work for you) or switch to the non-free CKEditor 4 LTS package. See https://ckeditor.com/ckeditor-4-support/ for more on this. (Note! This notice has been added by the django-ckeditor developers and we are not affiliated with CKSource and were not involved in the licensing change, so please refrain from complaining to us. Thanks.)
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions, sites
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying main.0001_initial... OK
  Applying main.0002_menu_sociallink_remove_product_category_and_more... OK
  Applying main.0003_page_featured_image_alter_sociallink_icon_pageview_and_more... OK
  Applying main.0004_delete_role... OK
  Applying main.0005_alter_pageview_referrer... OK
  Applying sessions.0001_initial... OK
  Applying sites.0001_initial... OK
  Applying sites.0002_alter_domain_unique... OK

```

Рисунок 3.19. Використання команди python manage.py runserver, яка активує веб-сервер Django

Головна сторінка повинна містити привітальний банер з текстом "Ласкаво просимо до CMS", а також кнопку "Зв'яжіться з нами" (рисунок 3.20). Оскільки цей інтерфейс розроблений з використанням шаблонів HTML і стилів CSS, тестується також їх коректність. Текст має бути видимим і відображатися правильно, без помилок, а стилі (кольори, шрифти, розміри елементів) повинні відповідати вимогам дизайну.

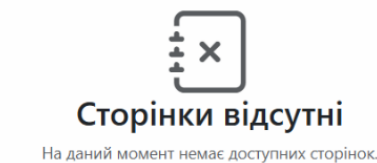
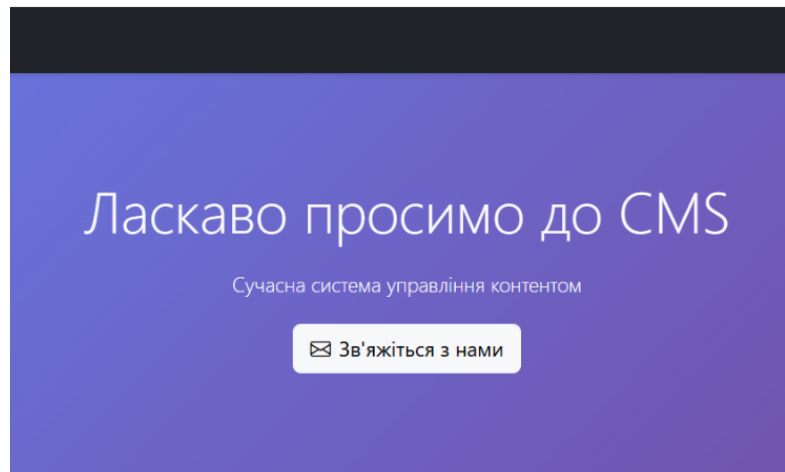


Рисунок 3.20. Головна сторінка

Тестування адаптивності: важливо перевірити, чи коректно адаптується головна сторінка до різних пристроїв, таких як мобільні телефони, планшети та ноутбуки. Веб-сайт має коректно відображатися на різних екранах, адаптуючи елементи під ширину дисплею.

Тестування кнопок і посилань: перевірка того, чи правильно працюють кнопки та посилання на головній сторінці. Користувач має бути перенаправлений на сторінку зворотного зв'язку після натискання на кнопку "Зв'яжіться з нами".

3.2.3. Перевірка навігації та переходів між сторінками

У розробленій системі передбачено навігаційне меню, яке дозволяє користувачам швидко переміщатися між різними сторінками сайту. При тестуванні цього етапу важливо перевірити, чи правильно відображаються пункти меню та чи працює маршрутизація. Потрібно перевірити, чи кожен пункт меню відображає відповідну сторінку після натискання.

Особливу увагу слід звернути на:

- Правильність URL-адрес: перевірка того, чи відповідають адреси сторінок правильним маршрутам на сервері.
- Коректне відображення активного пункту меню: візуальна індикація активного пункту меню, коли користувач перебуває на відповідній сторінці.
- Перевірка некоректних посилань: система повинна відображати повідомлення про помилку, якщо користувач переходить на неіснуючі сторінки або викликає помилку "404".

3.2.4. Тестування форми зворотного зв'язку

Форма зворотного зв'язку є важливим компонентом будь-якої CMS. Вона дає змогу користувачам надсилати повідомлення адміністраторам системи. Перевірка цієї форми включає кілька етапів, що подані в таблиці 3.1.

Таблиця 3.1

Етапи тестування форми зворотного зв'язку

№	Етапи	Опис
1	Заповнення полів	перевіряються сценарії, коли користувач залишає одне або декілька полів пустими або вводить некоректні дані, наприклад, невірний формат електронної пошти. Система повинна коректно обробляти ці помилки і надавати відповідні повідомлення
2	Відправлення форми	якщо форма заповнена правильно, при натисканні кнопки "Надіслати" користувач має побачити повідомлення про успішну відправку, а також бути перенаправленим на сторінку підтвердження
3	Перевірка обробки даних на сервері	після відправлення повідомлення перевіряється, чи потрапили дані в базу даних або чи було надіслано електронного листа адміністратору

3.2.5. Тестування багатомовності системи

Для малого бізнесу, який може мати клієнтів з різних країн, підтримка кількох мов є важливим компонентом. У CMS мають бути передбачені функціональні можливості для перемикання мови інтерфейсу (рисунок 3.21). Тестування багатомовності передбачає перевірку:

- Коректності зміни мови: при виборі іншої мови всі текстові елементи інтерфейсу, включаючи заголовки, кнопки, інструкції, повинні автоматично відображатися на новій мові.
- Перевірки правильності перекладів: необхідно перевірити, чи є помилки в перекладі на різні мови (наприклад, на українську чи англійську).

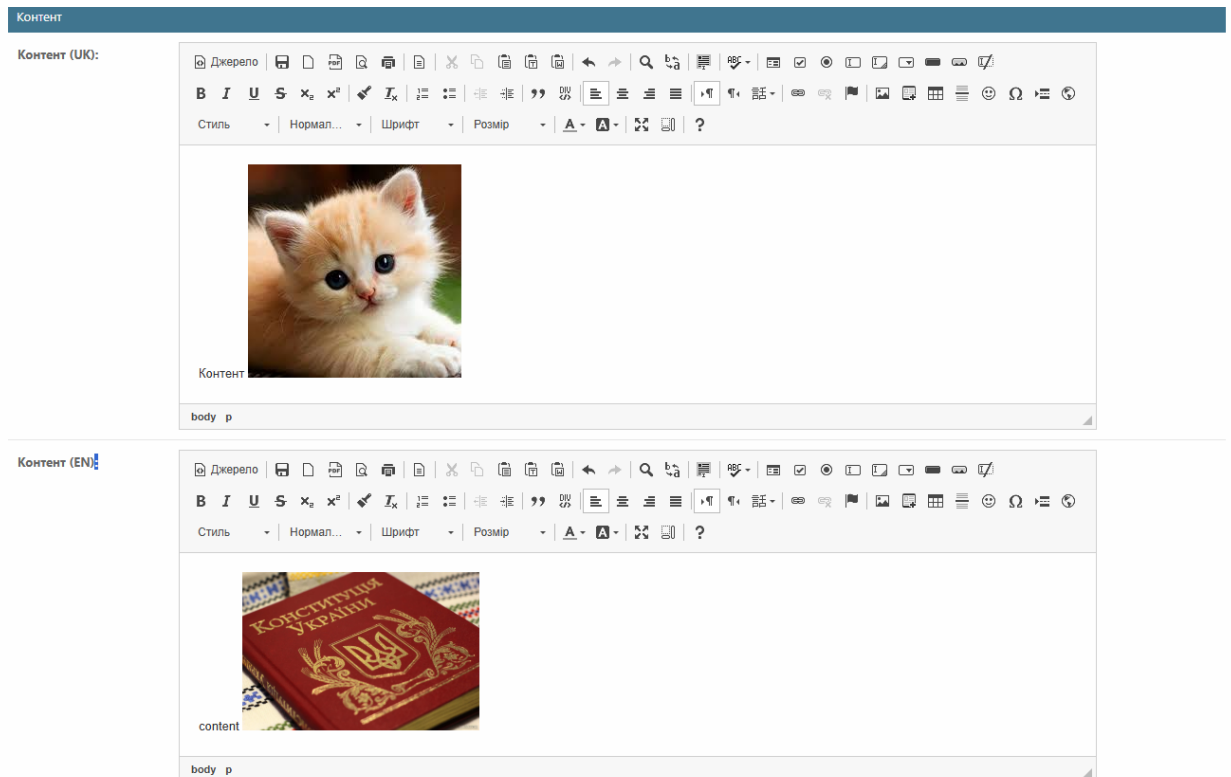


Рисунок 3.21. Додавання контенту українською та англійською мовами

3.2.6. Тестування відображення динамічного контенту

Один із головних елементів роботи CMS — це можливість створення динамічних сторінок. Користувачі можуть додавати текст, зображення, мета-теги та інший контент на сторінки через адміністративну панель. Це тестується шляхом:

- Створення нових сторінок (рисунки 3.22): перевіряється, чи можна додавати сторінки з різними типами контенту, такими як текст, зображення, посилання на інші сторінки (рисунки 3.23).

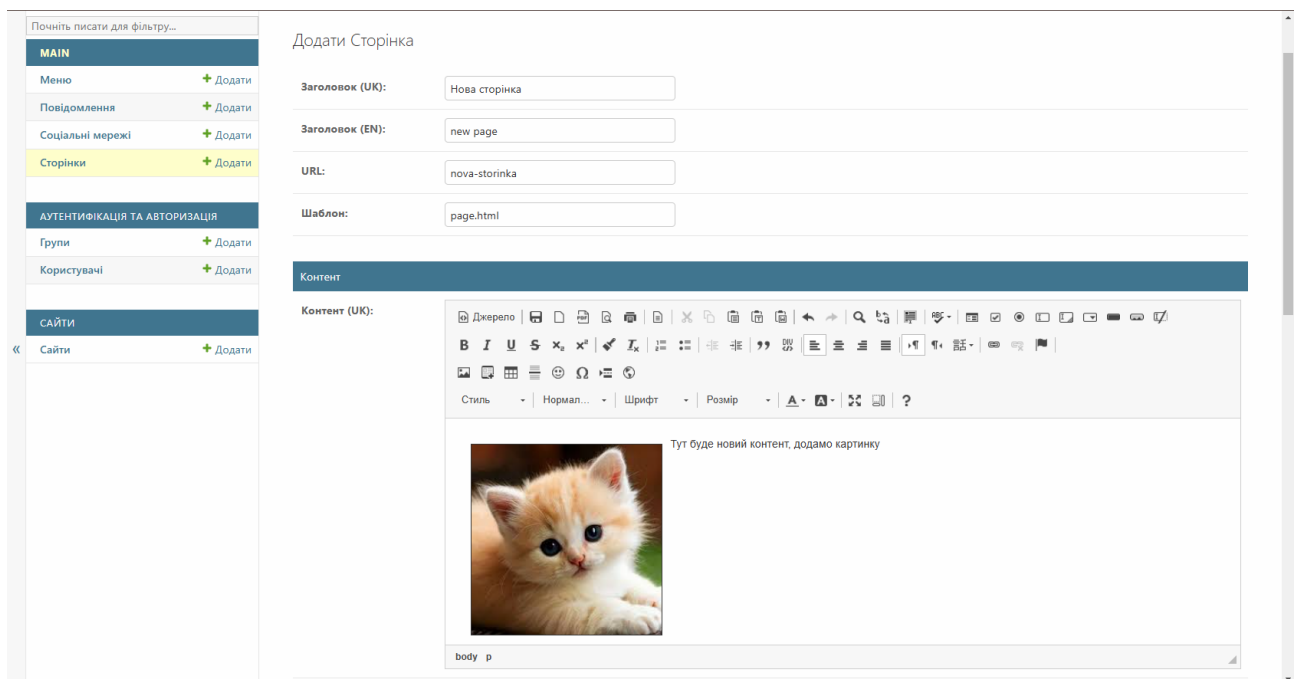


Рисунок 3.22. Створення нової сторінки

Властивості зображення

Інформація про зображення

Посилання

Надіслати

Додаткове

URL*

/media/uploads/admin/2025/01/20/images.jpg

Огляд Сервера

Альтернативний текст

Ширина

225

Висота

225

Рамка

Гориз. відступ

Верт. відступ

Вирівнювання

<не визна' ▾

🔒 ↺

Попередній перегляд



ОК

Скасувати

Рисунок 3.23. Властивості зображення

– Правильне відображення контенту на сайті (рисунок 3.24, рисунок 3.25): після публікації нової сторінки перевіряється, чи правильно відображаються тексти, зображення, відео та інші медіа. Важливо, щоб сторінка виглядала правильно як на настільних комп'ютерах, так і на мобільних пристроях.

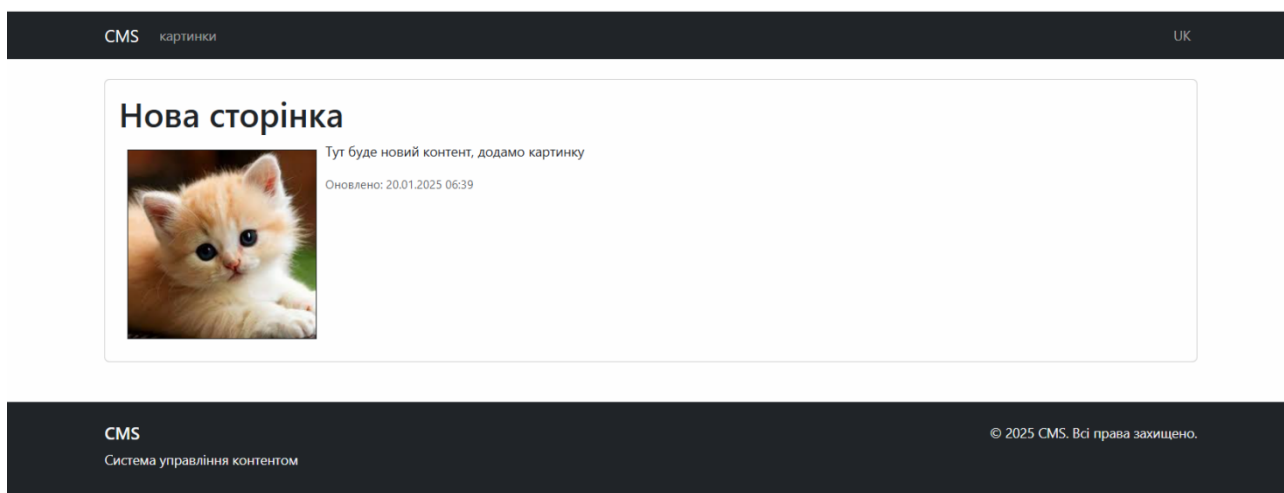


Рисунок 2.24. Відображення контенту (зображення)

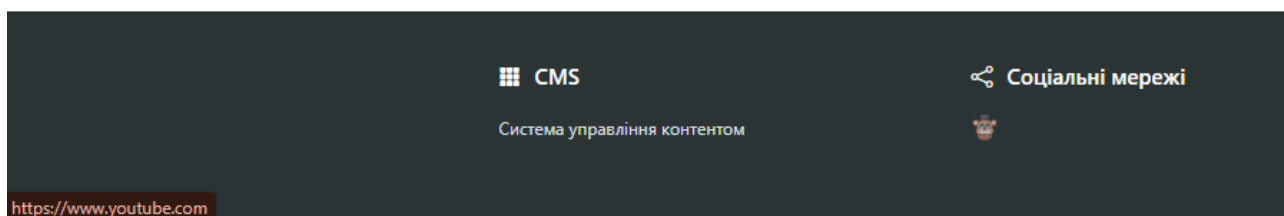


Рисунок 3.25. Відображення доданих соціальних мереж

– Метаінформація та SEO (рисунок): перевірка того, чи коректно відображаються мета-дані сторінки, наприклад, мета-теги для заголовку, опису та ключових слів, що покращують видимість сторінки в пошукових системах.

3.2.7. Тестування навантаження та стабільності

Тестування навантаження передбачає перевірку того, як система працює при високій кількості одночасних запитів. Для цього використовуються інструменти для навантажувального тестування, які можуть створити велику кількість одночасних запитів до сервера, імітуючи багатьох користувачів (додавання, зміна та права користувачів – рисунок 3.26, рисунок 3.27, рисунок 3.28, рисунок 3.29). Важливо, щоб система не давала збоїв, забезпечувала швидке завантаження сторінок та коректно обробляла запити навіть при високому навантаженні.

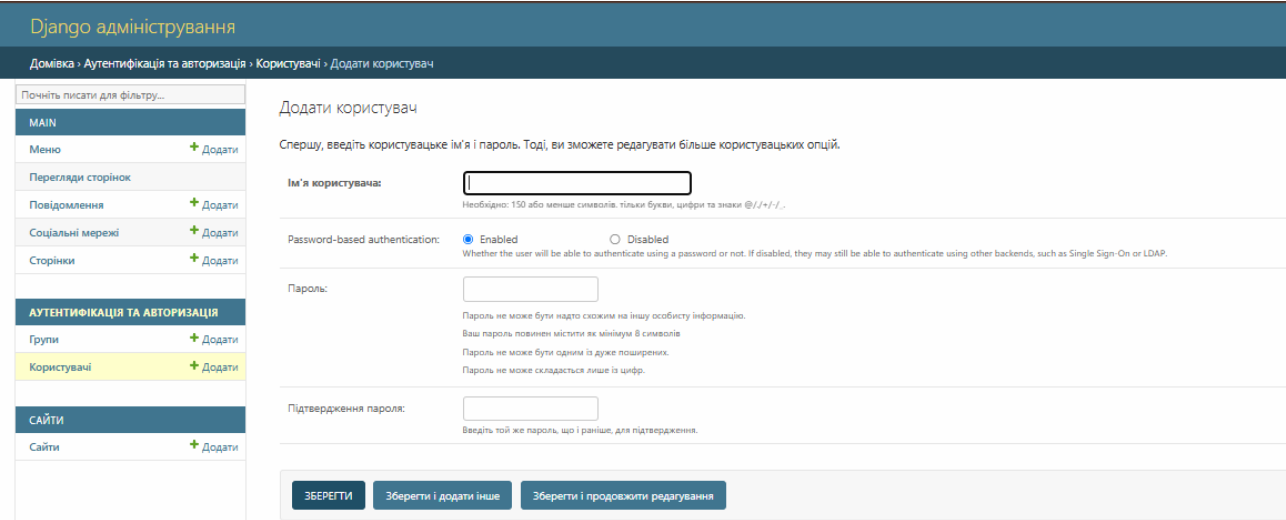


Рисунок 3.26. Додавання користувача

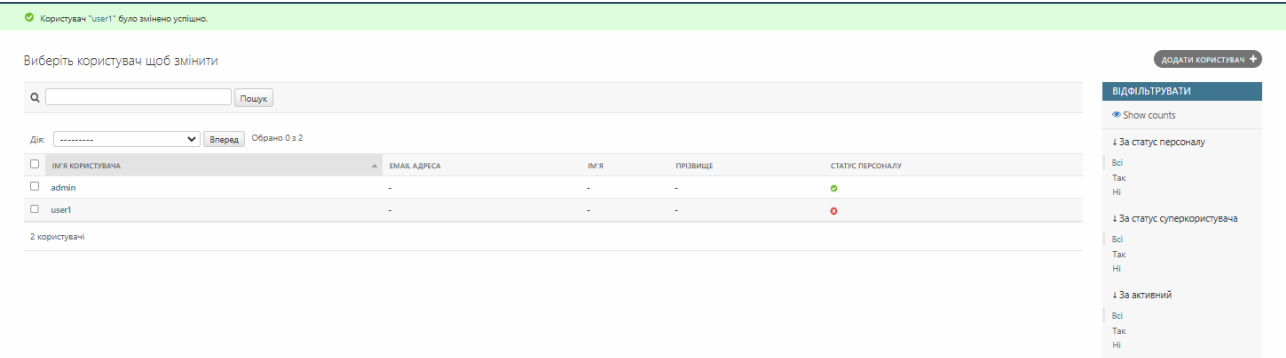


Рисунок 3.27. Вибір користувача для зміни

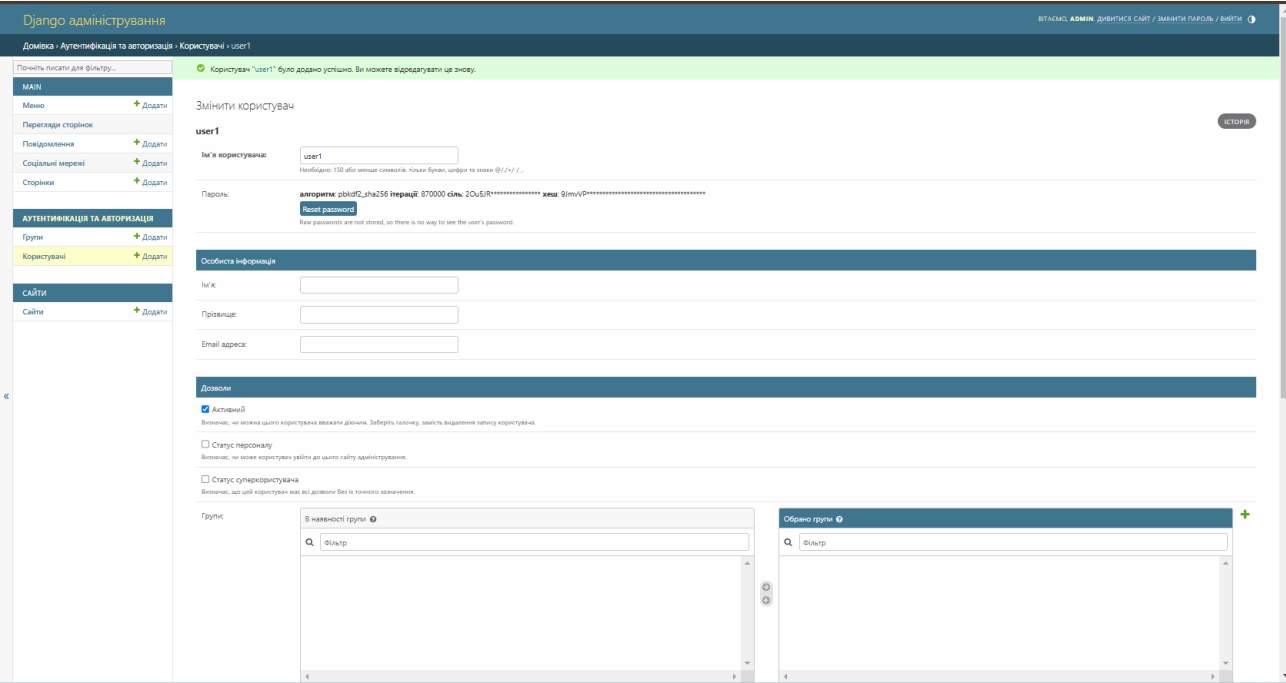


Рисунок 3.28. Зміна (редагування) користувача

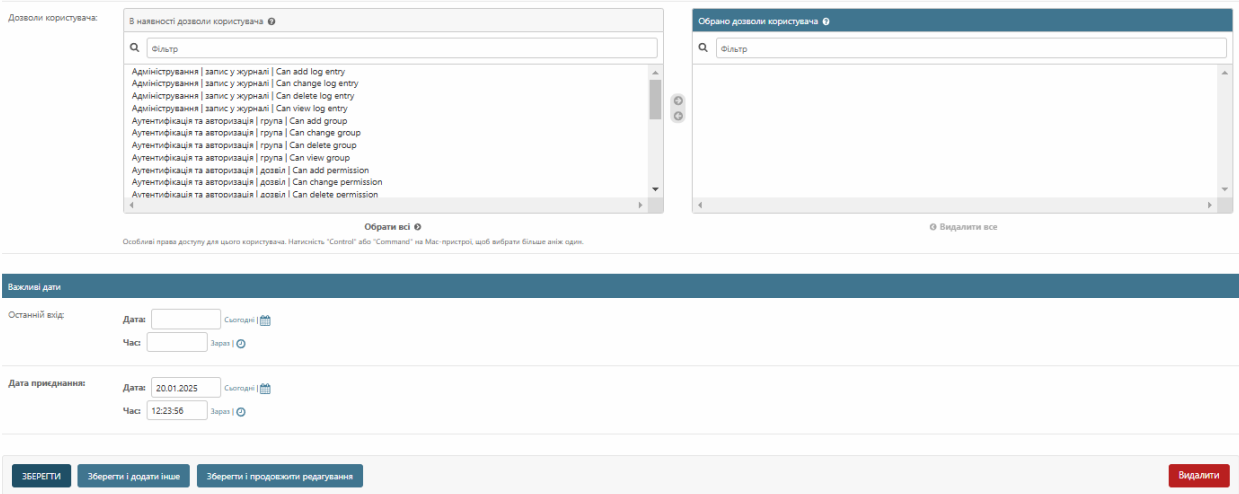


Рисунок 3.29. Дозволи користувача

Швидкість завантаження: тестується, чи швидко завантажуються сторінки за різних умов.

Надійність роботи: перевіряється стабільність роботи системи під високим навантаженням.

Ретельне тестування системи управління контентом є невід'ємною частиною процесу її розробки. Кожен етап тестування — від перевірки головної сторінки до забезпечення стабільної роботи під навантаженням — підтверджує, що система є надійною, безпечною та зручною для користувачів. Розроблена CMS готова до використання малим бізнесом, оскільки забезпечує всі основні функціональні можливості для управління контентом і взаємодії з клієнтами. Вдосконалене тестування гарантує, що система працюватиме коректно та ефективно на всіх етапах її використання.

3.3. Проблеми та перспективи вдосконалення CMS

Системи управління контентом відіграють ключову роль у створенні та підтримці вебресурсів для малого бізнесу. Вони дозволяють власникам підприємств зосередитися на своїй основній діяльності, автоматизуючи процеси публікації та адміністрування вебсайтів. Проте ефективність таких систем часто стикається з низкою викликів, які обмежують їхній потенціал.

Зростаючі потреби користувачів, стрімкий розвиток технологій, підвищення стандартів безпеки та вимоги до зручності використання визначають необхідність безперервного вдосконалення таких платформ. У рамках цього аналізу будуть розглянуті ключові труднощі, з якими стикаються під час розробки та використання CMS для малого бізнесу, а також запропоновані шляхи їх вирішення та перспективи подальшого розвитку.

3.3.1. Проблеми, що виникають у процесі розробки та використання CMS

Розробка систем управління контентом передбачає вирішення складного комплексу завдань, кожне з яких є критично важливим для забезпечення стабільної роботи платформи. Однією з головних проблем залишається питання забезпечення безпеки. CMS, що працюють у відкритому доступі, часто стають мішенню для кібератак. Зловмисники шукають вразливі місця в системах, особливо в тих, що використовують популярні фреймворки з відкритим кодом. Без належного рівня захисту, включно з перевіркою введених даних, впровадженням захищеного протоколу HTTPS та іншими засобами запобігання атакам, безпека системи може бути скомпрометована, що загрожує не лише роботі вебресурсу, а й довірі клієнтів.

Продуктивність також залишається важливим викликом, особливо для бізнесів, які мають обмежені фінансові ресурси для утримання високопродуктивних серверів. Велика кількість одночасних запитів або громіздкі бази даних можуть уповільнити роботу сайту, особливо у періоди пікового навантаження. Ускладнення виникають, коли необхідно інтегрувати сторонні бібліотеки чи послуги, які можуть сповільнювати роботу системи через конфлікти або неефективну оптимізацію коду.

Ще однією серйозною проблемою є інтеграція CMS із платформами, які часто використовуються малим бізнесом. Багато підприємств, зокрема інтернет-магазини, застосовують CRM-системи, сервіси електронної пошти чи платіжні інструменти. Несумісність API або обмежена документація до них

ускладнює злагоджену роботу таких платформ із CMS. Це створює додаткові витрати часу та ресурсів на пошук обхідних шляхів для інтеграції.

Користувацький досвід також може бути під загрозою через надмірну складність системи. Відсутність інтуїтивно зрозумілого інтерфейсу стає на заваді ефективному управлінню контентом. Особливо це стосується малого бізнесу, де персонал не завжди має технічні знання. Підприємці очікують, що система буде простою у використанні, але часто стикаються з непрозорістю процесів налаштування чи публікації контенту.

Масштабованість є ще одним питанням, яке потребує уваги. Для багатьох малих бізнесів, які з часом розширюють свої послуги чи аудиторію, стає важливим забезпечення стабільної роботи вебсайту при зростанні навантаження. Якщо система не може забезпечити адаптацію до нових обсягів даних, продуктивність суттєво знижується, що негативно впливає на довіру користувачів і ефективність бізнесу загалом.

Додатковим ускладненням виступає робота з багатомовним контентом. Для бізнесу, що обслуговує клієнтів із різних регіонів, важливо забезпечити коректний переклад і локалізацію сайту. Проте багато платформ не надають достатньо інструментів для гнучкої роботи з кількома мовами, що може спричиняти помилки або затримки в публікації контенту.

3.3.2. Перспективи вдосконалення CMS

Розвиток систем управління контентом може спрямовуватися на вирішення вищезазначених проблем. Одним із ключових напрямків є посилення заходів безпеки. Інтеграція автоматичних систем моніторингу та сканування вразливостей дозволить виявляти потенційні ризики до того, як вони вплинуть на роботу системи. Використання двофакторної аутентифікації або інших інноваційних методів захисту сприятиме підвищенню довіри з боку користувачів.

Для оптимізації продуктивності можна впроваджувати механізми кешування, які зменшують навантаження на бази даних, а також застосовувати

технології CDN (Content Delivery Network), що забезпечують швидку доставку контенту навіть для віддалених користувачів. Додатково CMS може пропонувати інструменти для автоматичної оцінки швидкості сайту з наданням рекомендацій щодо її покращення.

Щоб спростити інтеграцію з іншими платформами, необхідно розвивати стандартизовані API, які забезпечують взаємодію між CMS та сторонніми сервісами. Використання універсальних конекторів, що дозволяють підключати CRM, платіжні системи або маркетингові платформи, полегшить роботу підприємств, які використовують кілька інструментів одночасно.

Перспективним є впровадження інструментів для роботи з багатомовним контентом. Розширення функціоналу локалізації, автоматичне виявлення мовного контексту, а також інтеграція з сервісами машинного перекладу можуть значно спростити завдання створення багатомовних вебсайтів.

З огляду на зростання мобільного трафіку, важливо зосередити увагу на забезпеченні адаптивного дизайну. CMS має пропонувати шаблони, які автоматично підлаштовуються до розмірів екранів різних пристроїв, забезпечуючи комфортний перегляд контенту.

Інтеграція штучного інтелекту відкриває нові можливості для управління контентом. Алгоритми можуть аналізувати поведінку користувачів, генерувати персоналізовані рекомендації чи автоматизувати створення SEO-оптимізованих текстів. Такі інструменти спрощують роботу з системою та підвищують її ефективність.

Для забезпечення довговічності платформ важливо впровадити механізми автоматичного оновлення. CMS має не лише оновлюватися без потреби ручного втручання, але й забезпечувати резервне копіювання даних, що гарантує їхню збереженість у разі непередбачуваних обставин.

Сучасні системи управління контентом є багатофункціональними платформами, що значно спрощують адміністрування вебресурсів для малого бізнесу. Проте ефективність таких систем залежить від їхньої здатності реагувати на виклики безпеки, продуктивності, зручності використання та

масштабованості. Впровадження нових технологій, таких як штучний інтелект, автоматизація процесів і покращення роботи з багатомовним контентом, відкриває широкі перспективи для вдосконалення CMS. Завдяки зусиллям розробників платформи можуть перетворитися з інструментів для управління контентом на комплексні рішення, що сприяють стратегії розвитку бізнесу в умовах цифрової епохи.

Висновки до розділу 3

Розробка і тестування системи управління контентом (CMS) є критично важливими етапами створення сучасної платформи для малого бізнесу. Це складний процес, який охоплює широкий спектр завдань — від проектування базової структури системи до впровадження передових функціональних модулів і забезпечення їх належної взаємодії. У результаті цього процесу формується надійний інструмент, що дозволяє ефективно адмініструвати контент, підтримувати якісний зворотний зв'язок із користувачами та забезпечувати стабільну роботу вебсайту в умовах реального навантаження.

Процес створення основних модулів передбачає ретельне продумування їх функціональності відповідно до потреб малого бізнесу. Зокрема, модуль авторизації забезпечує безпеку даних і контроль доступу до системи, що є фундаментальним для будь-якої платформи, орієнтованої на комерційне використання. Управління сторінками є не менш важливим елементом, адже завдяки цьому модулю підприємці мають змогу створювати, редагувати та організовувати інформацію на своєму сайті з мінімальними витратами часу. Інтеграція мультимедійних інструментів дозволяє збагачувати контент візуальними матеріалами, які сприяють кращій взаємодії з аудиторією, а функції SEO забезпечують підвищення видимості сайту у пошукових системах, що є життєво необхідним для залучення нових клієнтів. Усі ці елементи взаємодіють між собою, створюючи цілісну екосистему, яка відповідає потребам сучасного бізнесу.

Ретельне тестування системи управління контентом гарантує її стабільну і коректну роботу. Цей етап є вирішальним для визначення того, наскільки надійно працюють основні модулі, і чи відповідає система очікуванням майбутніх користувачів. Зокрема, тестування дозволяє виявити слабкі місця у функціональності, продуктивності чи сумісності, що особливо важливо для малого бізнесу, де кожен збій може призвести до фінансових втрат або втрати довіри клієнтів. Перевірка всіх можливих сценаріїв використання платформи —

від роботи під високим навантаженням до взаємодії із зовнішніми системами — дозволяє гарантувати її ефективність у реальних умовах. Це також забезпечує впевненість, що система зможе адаптуватися до змін потреб бізнесу, наприклад, при розширенні клієнтської бази чи зростанні обсягів контенту.

Водночас розробка та тестування CMS висвітлює певні труднощі, які потребують подальшої роботи. Серед таких викликів можна виокремити питання безпеки, адже платформи, які використовуються для адміністрування вебресурсів, є потенційно вразливими до кібератак. Забезпечення захисту від таких загроз вимагає постійного оновлення системи, інтеграції сучасних протоколів шифрування даних і впровадження багатофакторної аутентифікації. Продуктивність також залишається одним із критичних чинників успішної роботи CMS, особливо для малих бізнесів, які обмежені в ресурсах. Впровадження технологій кешування, оптимізації баз даних та використання мереж доставки контенту (CDN) є важливими кроками для покращення швидкості завантаження сайту та зменшення часу відгуку системи.

Не менш значущим є питання зручності використання платформи. Простота інтерфейсу та інтуїтивно зрозуміле управління контентом є ключовими факторами, що визначають успішність CMS серед користувачів. Для підприємців, які не мають значного технічного досвіду, система повинна бути максимально зрозумілою, а навчання її використанню — мінімальним. Забезпечення високого рівня користувацького досвіду вимагає не лише якісного дизайну, а й продуманих механізмів навігації, автоматизації повторюваних процесів і адаптивності до різних пристроїв.

Варто звернути увагу й на питання масштабованості. Малі бізнеси, які обирають CMS, часто сподіваються на те, що їхній сайт зможе розвиватися разом із компанією. Це передбачає необхідність забезпечення гнучкості платформи, яка дозволяє легко додавати нові функції, розширювати базу даних або інтегрувати сторонні сервіси. Для цього необхідно орієнтуватися на модульну архітектуру, що полегшує процес оновлення та модифікації системи.

Водночас перспективи розвитку CMS відкривають нові горизонти. Інтеграція штучного інтелекту, автоматизація процесів створення контенту та впровадження інструментів аналітики дозволять значно підвищити ефективність платформи. Наприклад, автоматизовані рекомендації щодо оптимізації SEO чи аналіз поведінки користувачів сайту можуть сприяти кращій адаптації контенту до потреб аудиторії. Покращення функцій для роботи з багатомовним контентом також є важливим напрямком розвитку, адже це дозволить малим бізнесам охоплювати міжнародні ринки.

Таким чином, система управління контентом для малого бізнесу є не лише інструментом для створення та адміністрування вебсайту, а й стратегічним ресурсом, який дозволяє компаніям ефективніше взаємодіяти з аудиторією, підвищувати свою конкурентоспроможність та адаптуватися до динамічних змін у бізнес-середовищі. Розробка основних модулів, ретельне тестування та аналіз проблем, що виникають, створюють міцну основу для подальшого вдосконалення таких платформ. Завдяки постійній роботі над усуненням недоліків та впровадженню інноваційних рішень CMS стають невіддільною частиною сучасного цифрового бізнесу.

ВИСНОВКИ

У процесі розробки системи управління контентом для малого бізнесу реалізовано багаторівневий підхід, що поєднує сучасні технології з глибоким розумінням потреб цільової аудиторії. Основною метою цього проєкту було створення CMS, яка б задовольняла запити малого бізнесу завдяки простоті використання, доступності, адаптивності та розширеним функціональним можливостям. Розробка проводилася з урахуванням існуючих рішень на ринку, їхніх сильних сторін та недоліків, що дозволило обґрунтовано вибрати оптимальні технічні засоби для реалізації системи.

Дослідження починалося з аналізу ринкових потреб малого бізнесу, що стало ключовим у визначенні пріоритетних функціональних можливостей системи. Одним із основних викликів стало досягнення балансу між багатством функціоналу та зручністю його використання для користувачів без спеціалізованої технічної підготовки. Було враховано, що малий бізнес часто стикається з фінансовими обмеженнями та браком професійних ресурсів, тому важливою вимогою стала простота інтерфейсу, що мінімізує час на навчання персоналу та адміністрування сайту. У ході дослідження було проаналізовано популярні CMS, серед яких WordPress, Joomla, Drupal, Wix, OpenCart, Shopify та MODX. Ці системи мали як переваги, так і недоліки, які були використані для формування вимог до розроблюваної системи. Наприклад, складність адаптації та надмірна функціональність деяких платформ стали контрастом до бажання створити інтуїтивно зрозуміле рішення для кінцевого користувача.

Особливу увагу приділено технічним елементам розробки, зокрема вибору архітектури системи. Основними принципами, що закладалися у проєкт, були модульність, гнучкість та масштабованість. Створена архітектура дозволяє легко додавати нові функції, адаптувати систему до зростання бізнесу або змінювати окремі модулі без ризику порушення її цілісності. У процесі проєктування враховано сучасні вимоги до безпеки, що є важливим фактором в умовах збільшення кількості кіберзагроз. Передбачено багаторівневий захист

даних, що включає безпечну авторизацію, запобігання атакам через форму введення даних та шифрування конфіденційної інформації.

Система управління контентом передбачає широкий набір функцій, які відповідають потребам малого бізнесу. Серед них модулі для створення та редагування сторінок, управління мультимедіа, формування навігації, SEO-оптимізації та мультимовності. Інтеграція можливостей пошукової оптимізації забезпечує покращення видимості бізнесу в пошукових системах, що є ключовим у сучасних умовах, де онлайн-присутність відіграє важливу роль у залученні клієнтів. Крім того, реалізована підтримка багатомовності дозволяє бізнесу ефективно працювати з клієнтами різних мовних груп, що є особливо актуальним у багатонаціональних регіонах або для компаній, які прагнуть розширити свою аудиторію на міжнародному рівні. Важливим компонентом системи є модуль зворотного зв'язку, який забезпечує взаємодію з клієнтами та створює основу для побудови довготривалих ділових відносин.

Реалізація інтерфейсу користувача проводилася з акцентом на зручність та інтуїтивність. Інтерфейс побудовано за принципами адаптивного дизайну, що дозволяє забезпечити коректне відображення контенту на різних пристроях. Це особливо важливо в умовах, коли значна частина користувачів здійснює доступ до вебсайтів через мобільні пристрої. Використання сучасних бібліотек і фреймворків у розробці інтерфейсу дозволило досягти високої швидкодії та привабливості зовнішнього вигляду.

Одним із ключових етапів проєкту стало тестування розробленої системи. Було перевірено працездатність усіх функціональних модулів, коректність навігації, відображення контенту та роботи мультимовності. Окрему увагу приділено тестуванню навантаження та стабільності системи, що дало змогу оцінити її продуктивність за умов реального використання. Проведено перевірку форм зворотного зв'язку, динамічного контенту, а також інтеграції з аналітичними інструментами. Результати тестування підтвердили відповідність системи заявленим вимогам, її надійність та готовність до впровадження у практичну діяльність.

Процес розробки також виявив низку проблем, які мали бути вирішені для досягнення поставлених цілей. Серед них – складність забезпечення сумісності з різними браузерями та пристроями, а також необхідність оптимізації продуктивності системи для роботи з великими обсягами даних. Однак ці виклики було подолано завдяки використанню сучасних технологій та інструментів для тестування й оптимізації. У результаті отримано рішення, яке не тільки задовольняє потреби малого бізнесу, але й має потенціал для подальшого вдосконалення та розширення функціоналу.

Розроблена система управління контентом може бути легко інтегрована у різні бізнес-процеси та адаптована до специфічних потреб конкретного підприємства. Вона дозволяє значно скоротити витрати на створення та обслуговування вебресурсів, забезпечує гнучкість у роботі та високу якість взаємодії з клієнтами. Крім того, розробка цього проєкту відкриває нові можливості для досліджень у галузі управління контентом, зокрема у напрямку інтеграції з технологіями штучного інтелекту, автоматизації процесів та підвищення рівня персоналізації взаємодії з користувачами.

Підсумовуючи, розробка системи управління контентом для малого бізнесу підтвердила доцільність створення спеціалізованих рішень, які враховують особливості цільової аудиторії та сучасні вимоги до функціональності, безпеки й доступності. Отримані результати є вагомим внеском у розвиток інформаційних технологій для малого бізнесу та демонструють перспективність подальших досліджень у цій сфері.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Н. В. Попова, А. В. Катаєв, Л. В. Базалієва, О. І. Кононов, Т. А. Муха. МАРКЕТИНГОВІ КОМУНІКАЦІЇ Підручник Під загальною редакцією Н. В. Попової. Харків – 2020
2. A deep dive into Content Management Systems: 10 business benefits you should know URL: <https://www.agilitypr.com/pr-news/public-relations/a-deep-dive-into-content-management-systems-10-business-benefits-you-should-know/>
3. What is a content management system (CMS)? URL: <https://www.ibm.com/think/topics/content-management-system>
4. Martin Aranovitch. Content Management 101: A Practical Blueprint For Small Businesses. 2024
5. Правила інформаційної доступності для сайту URL: <https://www.vplyv.org.ua/archives/8225>
6. Michelle Hawley, Scott Clark. What Is a Content Management System (CMS)?. 2024
7. Як утримати клієнтів і підвищити їхню лояльність до вашого бізнесу URL: <https://business-broker.com.ua/blog/iak-utrymaty-kliientiv-i-pidvyshchyty-ikhniu-loialnist-do-vashoho-biznesu/>
8. Learn about content management — definition, benefits, and more URL: <https://business.adobe.com/blog/basics/content-management>
9. Ireneusz Żuchowski, Zoia Sharlovych, Olena Dudnyk. TRANSFORMACJE CYFROWE I TECHNOLOGIE INNOWACYJNE W EKONOMII. ZBIÓR MATERIAŁÓW Z MIĘDZYNARODOWEJ NAUKOWO-PRAKTYCZNEJ KONFERENCJI INTERNETOWEJ. Część 2. Łomża-Charków. 14-15.03.2024
10. Перевірка адаптивності сайту за допомогою браузера URL: <https://webtune.com.ua/statti/internet-marketing/yak-pereviraty-adaptyvnist-za-dopomogoyu-brauzera/>

11. Адаптивний сайт: актуально та ефективно URL:
<https://asabix.com.ua/adaptive-website-relevant-and-effective/>
12. Вибір CMS для інтернет-магазину URL:
<https://guildofmarketing.com/vybir-cms-dlya-internet-magazynu/>
13. Стефан Рефельд. Захист даних та інформаційна безпека - за стандартами ISO 27001 та ISO 27701. 2022
14. Andrii Kostashchuk. What is a content management system (CMS)?. 2024
15. What is a Cloud CMS: Everything You Need to Know URL:
<https://www.dotcms.com/blog/what-is-a-cloud-cms>
16. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В БІЗНЕСІ. Частина 1. За заг. редакцією І.Б. Шевчук. Навчальний посібник. Львів. 2020
17. Ujjawal Laddha. What is a CMS? A Beginner's Guide to Content Management Systems. 2025
18. Найкращі інструменти SEO для малого бізнесу URL:
<https://bizzzdev.com/best-seo-tools-for-small-businesses/>
19. Aesha Sdeek Shaheen. Maximizing Website Performance with Google Analytics. Turkish Journal of Computer and Mathematics Education (TURCOMAT) 14(03):1273-1277. 2023
20. Що таке CMS? URL: <https://shelfy.com.ua/newsroom/shho-take-cms/>
21. WordPress Without Limits URL: <https://wordpress.com/>
22. Переваги та недоліки WordPress URL: <https://infosoft.ua/ua/perevahy-i-nedoliky-wordpress>
23. Переваги та недоліки CMS WordPress URL:
<https://cityhost.ua/uk/blog/preimuschestva-i-nedostatki-cms-wordpress.html>
24. Розробка інтернет-магазину на WordPress URL:
<https://webnauts.pro/uk/blog/rozrobka-internet-magazynu-na-wordpress/>
25. Joomla! URL: <https://www.joomla.org/>
26. Ключові переваги CMS Joomla URL:
<https://cityhost.ua/uk/blog/klyuchevye-preimuschestva-cms-joomla.html>

27. Огляд CMS Joomla: що це, плюси та мінуси URL: <https://interkassa.com/blog/oglyad-cms-joomla-shho-ce-plyusi-ta-minusi>
28. Benefits of Joomla Development URL: <https://medium.com/@Sterlingtechnolabs/benefits-of-joomla-development-6c02b4496175>
29. Create Ambitious Digital Experiences URL: <https://new.drupal.org/home>
30. 7 переваг створення сайту на Drupal URL: <https://smile-ukraine.com/ua/blog/7-benefits-drupal-website-development#risenna-z-vidkritim-kodom>
31. The pros and cons of Drupal URL: <https://www.techtarget.com/searchcontentmanagement/tip/The-pros-and-cons-of-Drupal>
32. Can Drupal 10 development services handle large-scale projects? URL: <https://medium.com/@thecommerceshop/can-drupal-10-development-services-handle-large-scale-projects-f670b6b6b2cb>
33. Create a website without limits URL: <https://www.wix.com/>
34. Розробка сайту на Wix - плюси та мінуси URL: <https://www.legitdigital.org/post/wix-plusy-ta-minusy>
35. Pros and Cons of Using Wix - 13 Factors to Consider URL: <https://www.enzuzo.com/blog/pros-and-cons-of-using-wix>
36. Can Small Businesses use Wix for Website Development? Learn All about Wix in One post! URL: <https://www.linkedin.com/pulse/can-small-businesses-use-wix-website-development-all-joomla-expert-bwxpc/>
37. The best FREE and open-source eCommerce platform URL: <https://www.opencart.com/>
38. OpenCart переваги і недоліки URL: <https://webbuilding.pro/ukr/blog/opencart-advantages-disadvantages>
39. OpenCart Review: Features and Pricing for Your Online Store URL: <https://www.neo.space/blog/opencart>

40. Globalna platforma handlowa URL:
<https://www.shopify.com/pl?country=pl&lang=pl>
41. What exactly are the advantages of using Shopify? URL:
<https://community.shopify.com/c/shopify-discussions/what-exactly-are-the-advantages-of-using-shopify/m-p/2198659>
42. The 5 biggest disadvantages of Shopify URL:
<https://www.growcode.com/blog/the-5-biggest-disadvantages-of-shopify/>
43. How Shopify Empowers Businesses with E-Commerce Customer Experience (CX) Solutions URL: <https://www.renaissance.io/journal/how-shopify-empowers-businesses-with-e-commerce-customer-experience-cx-solutions>
44. Revolutionize Your Web Experience URL: <https://modx.com/>
45. Переваги та недоліки CMS MODx URL:
<https://cityhost.ua/uk/blog/preimuschestva-i-nedostatki-cms-modx.html>
46. Advantages and disadvantages of CMS MODx URL:
<https://cityhost.ua/en/blog/preimuschestva-i-nedostatki-cms-modx.html>
47. A Comprehensive Step-by-Step Guide to Utilizing MODX for Creating Dynamic User Profiles URL: <https://moldstud.com/articles/p-a-comprehensive-step-by-step-guide-to-utilizing-modx-for-creating-dynamic-user-profiles>
48. Introduction to client-side frameworks URL:
https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Introduction
49. Які мови програмування треба знати для створення сайтів URL:
<https://foxminded.ua/movy-prohramuvannia-dlia-stvorennia-saitiv/>
50. Адаптивний дизайн: інновації та перспективи веб-розробки URL:
<https://bdut.co.ua/pro-nas/adaptyvnyy-dyzayn/>
51. Using JsonField(in models) with Django and SQLite URL:
<https://medium.com/@manz442001/using-jsonfield-in-models-with-django-and-sqlite-827670bca3eb>

52. Google Analytics 4: повний гайд з налаштування GA4 та швидкого переходу URL: <https://wezom.com.ua/ua/blog/google-analytics-4-povniy-gayd-z-nalashtuvannya>
53. Google Analytics 4: повний гайд з налаштування GA4 та швидкого переходу URL: <https://wezom.com.ua/ua/blog/google-analytics-4-povniy-gayd-z-nalashtuvannya>
54. Django форми URL: https://tutorial.djangogirls.org/uk/django_forms/

Лістинг програми

main.py

```
# Django imports
from django.apps import AppConfig
from django.db import models
from django.contrib import admin
from django.urls import path, include
from django.template.response import TemplateResponse
from django.conf import settings
from django.conf.urls.static import static
from django.conf.urls.i18n import i18n_patterns
from django.http import Http404
from django.core.management.base import BaseCommand
from django.core import serializers
from django import forms
from django.utils import translation
from django.utils.translation import gettext_lazy as _
from django.shortcuts import redirect
from django.contrib.sitemaps import Sitemap
from django.contrib.sitemaps.views import sitemap
from django.views.i18n import set_language
# Image handling imports
from PIL import Image
from io import BytesIO
from django.core.files.uploadedfile import
InMemoryUploadedFile
import sys
import os
from datetime import datetime
# CKEditor import
from ckeditor_uploader.fields import RichTextUploadingField
# Image optimization
class OptimizedImageField(models.ImageField):
    def __init__(self, *args, **kwargs):
        self.max_width = kwargs.pop('max_width', 1920)
        self.max_height = kwargs.pop('max_height', 1080)
        self.quality = kwargs.pop('quality', 85)
        super().__init__(*args, **kwargs)
    def pre_save(self, model_instance, add):
        file = getattr(model_instance, self.attname)
```

```

        if file and isinstance(file, InMemoryUploadedFile):
            image = Image.open(file)
            if image.mode in ("RGBA", "P"):
                image = image.convert("RGB")
            if image.width > self.max_width or image.height >
self.max_height:
                image.thumbnail((self.max_width,
self.max_height), Image.Resampling.LANCZOS)
                output = BytesIO()
                image.save(output, format='JPEG',
quality=self.quality, optimize=True)
                output.seek(0)
                file = InMemoryUploadedFile(
                    output,
                    'ImageField',
                    f"{file.name.split('.')[0]}.jpg",
                    'image/jpeg',
                    sys.getsizeof(output),
                    None
                )
                setattr(model_instance, self.attname, file)
        return getattr(model_instance, self.attname)

# Models
class Menu(models.Model):
    name = models.CharField(_('Назва меню'), max_length=100)
    slug = models.SlugField(_('Ідентифікатор'), unique=True)
    class Meta:
        verbose_name = _('Меню')
        verbose_name_plural = _('Меню')
    def __str__(self):
        return self.name
class MenuItem(models.Model):
    menu = models.ForeignKey(Menu, on_delete=models.CASCADE,
related_name='items')
    title_uk = models.CharField(_('Назва (UK)'),
max_length=100)
    title_en = models.CharField(_('Назва (EN)'),
max_length=100)
    url = models.CharField(_('URL'), max_length=200)
    page = models.ForeignKey('Page',
on_delete=models.SET_NULL, null=True, blank=True)
    parent = models.ForeignKey('self',
on_delete=models.CASCADE, null=True, blank=True)
    order = models.IntegerField(_('Порядок'), default=0)

```

```

class Meta:
    verbose_name = _('Пункт меню')
    verbose_name_plural = _('Пункти меню')
    ordering = ['order']
def __str__(self):
    return self.get_title()
def get_title(self, lang=None):
    if lang is None:
        lang = translation.get_language() or
settings.LANGUAGE_CODE
        return getattr(self, f'title_{lang}', self.title_uk)
# Fallback to Ukrainian title
class AnalyticsMixin:
    def get_views_count(self):
        return PageView.objects.filter(page=self).count()
    def get_unique_views_count(self):
        return
PageView.objects.filter(page=self).values('ip_address').distinct().count()
class Page(models.Model, AnalyticsMixin):
    title_uk = models.CharField(_('Заголовок (UK)'),
max_length=200)
    title_en = models.CharField(_('Заголовок (EN)'),
max_length=200)
    slug = models.SlugField(_('URL'), unique=True)
    content_uk = RichTextUploadingField(_('Контент (UK)'))
    content_en = RichTextUploadingField(_('Контент (EN)'))
    meta_title = models.CharField(_('МЕТА заголовок'),
max_length=200, blank=True)
    meta_description = models.TextField(_('МЕТА опис'),
blank=True)
    meta_keywords = models.CharField(_('МЕТА ключові слова'),
max_length=200, blank=True)
    template = models.CharField(_('Шаблон'), max_length=100,
default='page.html')
    featured_image = OptimizedImageField(
        _('Головне зображення'),
        upload_to='pages/',
        max_width=1200,
        max_height=800,
        quality=85,
        blank=True
    )

```

```

    is_active = models.BooleanField(_('Активна'),
default=True)
    created = models.DateTimeField(_('Створено'),
auto_now_add=True)
    modified = models.DateTimeField(_('Змінено'),
auto_now=True)
    class Meta:
        verbose_name = _('Сторінка')
        verbose_name_plural = _('Сторінки')
    def __str__(self):
        return self.get_title()
    def get_title(self, lang=None):
        if lang is None:
            lang = translation.get_language()
        return getattr(self, f'title_{lang}')
    def get_content(self, lang=None):
        if lang is None:
            lang = translation.get_language()
        return getattr(self, f'content_{lang}')
    def get_absolute_url(self):
        lang = translation.get_language()
        if lang != settings.LANGUAGE_CODE:
            return f'/{lang}/page/{self.slug}/'
        return f'/page/{self.slug}/'
class PageView(models.Model):
    page = models.ForeignKey(Page, on_delete=models.CASCADE)
    ip_address = models.GenericIPAddressField()
    user_agent = models.CharField(max_length=500)
    timestamp = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey('auth.User', null=True,
on_delete=models.SET_NULL)
    session_key = models.CharField(max_length=40)
    referrer = models.URLField(null=True, blank=True)
    class Meta:
        verbose_name = _('Перегляд сторінки')
        verbose_name_plural = _('Перегляди сторінок')
class SocialLink(models.Model):
    name = models.CharField(_('Назва'), max_length=100)
    url = models.URLField(_('URL'))
    icon = OptimizedImageField(
        _('Іконка'),
        upload_to='social/',
        max_width=64,
        max_height=64,

```

```

        null=True, # Разрешаем null значения
        blank=True # Разрешаем пустые значения
    )
    order = models.IntegerField(_('Порядок'), default=0)
    def save(self, *args, **kwargs):
        if self.icon:
            # Проверяем, существует ли директория
            upload_dir = os.path.join(settings.MEDIA_ROOT,
'social')
            if not os.path.exists(upload_dir):
                os.makedirs(upload_dir)
            # Если файл уже существует, создаем уникальное имя
            base_name = os.path.splitext(self.icon.name)[0]
            ext = os.path.splitext(self.icon.name)[1]
            counter = 1
            while
os.path.exists(os.path.join(settings.MEDIA_ROOT, 'social',
f"{base_name}{ext}")):
                base_name =
f"{os.path.splitext(self.icon.name)[0]}_{counter}"
                counter += 1
            self.icon.name = f"social/{base_name}{ext}"
            super().save(*args, **kwargs)
    class Meta:
        verbose_name = _('Соціальна мережа')
        verbose_name_plural = _('Соціальні мережі')
        ordering = ['order']
    def __str__(self):
        return self.name
    def icon_url(self):
        if self.icon:
            return self.icon.url
        return ''
class Feedback(models.Model):
    name = models.CharField(_('Ім\'я'), max_length=100)
    email = models.EmailField(_('Email'))
    message = models.TextField(_('Повідомлення'))
    created = models.DateTimeField(_('Створено'),
auto_now_add=True)
    processed = models.BooleanField(_('Оброблено'),
default=False)
    class Meta:
        verbose_name = _('Повідомлення')
        verbose_name_plural = _('Повідомлення')

```

```

        ordering = ['-created']
    def __str__(self):
        return f"{self.name} - {self.email}"
# Forms
class FeedbackForm(forms.ModelForm):
    class Meta:
        model = Feedback
        fields = ['name', 'email', 'message']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control'}),
            'email': forms.EmailInput(attrs={'class': 'form-control'}),
            'message': forms.Textarea(attrs={'class': 'form-control', 'rows': 5}),
        }
# Admin
class MenuItemInline(admin.TabularInline):
    model = MenuItem
    extra = 1
class MenuAdmin(admin.ModelAdmin):
    list_display = ['name', 'slug']
    inlines = [MenuItemInline]
class PageAdmin(admin.ModelAdmin):
    list_display = ['title_uk', 'title_en', 'slug', 'is_active', 'modified', 'get_views_count']
    list_filter = ['is_active']
    search_fields = ['title_uk', 'title_en', 'content_uk', 'content_en']
    prepopulated_fields = {'slug': ('title_uk',)}
    readonly_fields = ['created', 'modified']
    fieldsets = (
        (None, {
            'fields': ('title_uk', 'title_en', 'slug', 'template', 'featured_image')
        }),
        (_('КОНТЕНТ'), {
            'fields': ('content_uk', 'content_en')
        }),
        (_('SEO'), {
            'fields': ('meta_title', 'meta_description', 'meta_keywords')
        }),
    ),

```

```

        (_('Налаштування'), {
            'fields': ('is_active', 'created', 'modified')
        })),
    )
class SocialLinkAdmin(admin.ModelAdmin):
    list_display = ['name', 'url', 'order']
    list_editable = ['order']
class FeedbackAdmin(admin.ModelAdmin):
    list_display = ['name', 'email', 'created', 'processed']
    list_filter = ['processed']
    search_fields = ['name', 'email', 'message']
    readonly_fields = ['created']
class PageViewAdmin(admin.ModelAdmin):
    list_display = ['page', 'ip_address', 'user', 'timestamp']
    list_filter = ['page', 'timestamp']
    date_hierarchy = 'timestamp'
    def has_add_permission(self, request):
        return False
# Admin site registration
admin.site.register(Menu, MenuAdmin)
admin.site.register(Page, PageAdmin)
admin.site.register(SocialLink, SocialLinkAdmin)
admin.site.register(Feedback, FeedbackAdmin)
admin.site.register(PageView, PageViewAdmin)
# Sitemap
class PageSitemap(Sitemap):
    changefreq = 'weekly'
    priority = 0.5
    def items(self):
        return Page.objects.filter(is_active=True)
    def lastmod(self, obj):
        return obj.modified
# Middleware
class AnalyticsMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response
    def __call__(self, request):
        response = self.get_response(request)
        if hasattr(request, 'resolver_match') and
request.resolver_match is not None:
            view_name = request.resolver_match.view_name
            if view_name == 'page_detail':
                try:

```

```

        page_slug =
request.resolver_match.kwargs.get('slug')
        page = Page.objects.get(slug=page_slug)
        PageView.objects.create(
            page=page,
ip_address=request.META.get('REMOTE_ADDR'),
user_agent=request.META.get('HTTP_USER_AGENT', ''),
            user=request.user if
request.user.is_authenticated else None,
session_key=request.session.session_key or '',
referrer=request.META.get('HTTP_REFERER')
        )
    except Page.DoesNotExist:
        pass
    except Exception as e:
        print(f"Помилка при створенні запису
аналітики: {e}")
    return response
# Context processors
def languages_context(request):
    return {
        'languages': settings.LANGUAGES,
        'current_language': translation.get_language(),
    }
def get_common_context(request):
    return {
        'menus': Menu.objects.all(),
        'social_links': SocialLink.objects.all(),
        'current_language': translation.get_language(),
        'languages': settings.LANGUAGES,
    }
# Views
def home(request):
    context = get_common_context(request)
    context['pages'] =
Page.objects.filter(is_active=True).order_by('-created')
    return TemplateResponse(request, 'index.html', context)
def page_detail(request, slug):
    try:
        page = Page.objects.get(slug=slug, is_active=True)
    except Page.DoesNotExist:
        raise Http404(_('Сторінку не знайдено'))
    context = get_common_context(request)
    context['page'] = page

```

```

    return TemplateResponse(request, page.template, context)
def feedback(request):
    if request.method == 'POST':
        form = FeedbackForm(request.POST)
        if form.is_valid():
            form.save()
            return TemplateResponse(request,
'feedback_success.html', get_common_context(request))
        else:
            form = FeedbackForm()
            context = get_common_context(request)
            context['form'] = form
            return TemplateResponse(request, 'feedback.html', context)
# Backup command
class Command(BaseCommand):
    help = 'Створює резервну копію бази даних'
    def handle(self, *args, **options):
        backup_dir = 'backups'
        if not os.path.exists(backup_dir):
            os.makedirs(backup_dir)
        timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
        backup_file = os.path.join(backup_dir,
f'backup_{timestamp}.json')
        models_to_backup = [Page, Menu, MenuItem, SocialLink,
Feedback]
        with open(backup_file, 'w', encoding='utf-8') as f:
            for model in models_to_backup:
                serializers.serialize('json',
model.objects.all(), stream=f)
            self.stdout.write(
                self.style.SUCCESS(f'Резервну копію успішно
створено: {backup_file}')
            )
# URL конфігурація
app_name = 'main'
# Sitemap configuration
sitemaps = {
    'pages': PageSitemap,
}
# Base URL patterns (without language prefix)
base_urlpatterns = [
    path('admin/', admin.site.urls),
    path('ckeditor/', include('ckeditor_uploader.urls')),

```

```

    path('sitemap.xml', sitemap, {'sitemaps': sitemaps},
name='sitemap'),
    path('i18n/setlang/', set_language, name='set_language'),
]
# Language-specific URL patterns
language_patterns = [
    path('', home, name='home'),
    path('page/<slug:slug>/', page_detail,
name='page_detail'),
    path('feedback/', feedback, name='feedback'),
]
# Combine URL patterns
urlpatterns = base_urlpatterns + i18n_patterns(
    *language_patterns,
    prefix_default_language=False # Don't add prefix for
default language
)
# Add static/media serving in development
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
# App configuration
class MainConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'main'
    verbose_name = _('Головний застосунок')
    def ready(self):
        # Import signal handlers
        from . import signals

```

```
manage.py

# manage.py
#!/usr/bin/env python
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
        'settings')
    try:
        from django.core.management import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's
installed?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

settings.py

```
import os
from pathlib import Path
# Базова директорія проекту
BASE_DIR = Path(__file__).resolve().parent
# Налаштування безпеки
SECRET_KEY = os.environ.get('DJANGO_SECRET_KEY', 'your-secret-key-here')
DEBUG = os.environ.get('DJANGO_DEBUG', 'True') == 'True'
ALLOWED_HOSTS = os.environ.get('DJANGO_ALLOWED_HOSTS', '*').split(',')
# Конфігурація застосунків
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sitemaps',
    'django.contrib.sites',
    'ckeditor',
    'ckeditor_uploader',
    'main',
]
# Конфігурація посередників
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware', # Make sure
this is after session and before common
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'main.main.AnalyticsMiddleware', # Додано аналітики
]
# Налаштування URLs
ROOT_URLCONF = 'main.main'
# Налаштування шаблонів
TEMPLATES = [
    {
```

```

        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
                'main.main.languages_context', # Контекст для
MOV
            ],
        },
    ],
]
# Налаштування бази даних
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
# Валідатори паролів
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilari
tyValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidato
r',
        'OPTIONS': {
            'min_length': 8,
        }
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidat
or',
    },
    {

```

```

        'NAME':
        'django.contrib.auth.password_validation.NumericPasswordValida
tor',
    },
]
# Налаштування статичних файлів
STATIC_URL = 'static/'
STATIC_ROOT = BASE_DIR / 'static'
STATICFILES_DIRS = [
    BASE_DIR / 'static_dev',
]
# Налаштування медіа файлів
MEDIA_URL = 'media/'
MEDIA_ROOT = BASE_DIR / 'media'
# Налаштування локалізації
LANGUAGE_CODE = 'uk'
TIME_ZONE = 'Europe/Kiev'
USE_I18N = True
USE_TZ = True
LANGUAGES = [
    ('uk', 'Українська'),
    ('en', 'English'),
]
LOCALE_PATHS = [
    BASE_DIR / 'locale',
]
# Налаштування CKEditor
# CKEditor налаштування
CKEDITOR_UPLOAD_PATH = "uploads/"
CKEDITOR_JQUERY_URL =
'https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min
.js'
CKEDITOR_CONFIGS = {
    'default': {
        'toolbar': 'full',
        'height': 300,
        'width': '100%',
        'removePlugins': 'stylesheetparser',
        'allowedContent': True,
        'extraPlugins': ','.join([
            'div',
            'autolink',
            'autoembed',
            'embedsemantic',

```

```

        'autogrow',
        'widget',
        'lineutils',
        'dialog',
        'dialogui',
    ]),
},
}

CKEDITOR_BASEPATH = "/static/ckeditor/ckeditor/"
CKEDITOR_RESTRICT_BY_USER = True
CKEDITOR_IMAGE_BACKEND = "pillow"
# Налаштування для Sites framework
SITE_ID = 1
# Налаштування електронної пошти
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = os.environ.get('EMAIL_HOST', 'localhost')
EMAIL_PORT = os.environ.get('EMAIL_PORT', '25')
EMAIL_HOST_USER = os.environ.get('EMAIL_HOST_USER', '')
EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_HOST_PASSWORD',
    '')
EMAIL_USE_TLS = os.environ.get('EMAIL_USE_TLS', 'False') ==
    'True'
DEFAULT_FROM_EMAIL = os.environ.get('DEFAULT_FROM_EMAIL',
    'webmaster@localhost')
# Google Analytics
GA_TRACKING_ID = os.environ.get('GA_TRACKING_ID', '')
# Налаштування безпеки
SECURE_SSL_REDIRECT =
os.environ.get('DJANGO_SECURE_SSL_REDIRECT', 'False') ==
    'True'
SESSION_COOKIE_SECURE =
os.environ.get('DJANGO_SESSION_COOKIE_SECURE', 'False') ==
    'True'
CSRF_COOKIE_SECURE =
os.environ.get('DJANGO_CSRF_COOKIE_SECURE', 'False') == 'True'
SECURE_BROWSER_XSS_FILTER = True
SECURE_CONTENT_TYPE_NOSNIFF = True
X_FRAME_OPTIONS = 'DENY'
# Тип поля за замовчуванням
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
# Налаштування кешування
CACHES = {
    'default': {

```

```

        'BACKEND':
        'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': BASE_DIR / 'cache',
    }
}
# Налаштування для оптимізації зображень
IMAGE_UPLOAD_MAX_SIZE = {
    'thumbnail': (300, 300),
    'medium': (800, 800),
    'large': (1200, 1200),
}
# Налаштування для логування
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
            'format': '{levelname} {asctime} {module}
{message}',
            'style': '{',
        },
    },
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': BASE_DIR / 'debug.log',
            'formatter': 'verbose',
        },
        'console': {
            'level': 'INFO',
            'class': 'logging.StreamHandler',
            'formatter': 'verbose',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file', 'console'],
            'level': 'INFO',
            'propagate': True,
        },
        'main': {
            'handlers': ['file', 'console'],
            'level': 'INFO',

```

```

        'propagate': True,
    },
},
}
# Налаштування сесій
SESSION_ENGINE = 'django.contrib.sessions.backends.db'
SESSION_COOKIE_AGE = 1209600 # 2 тижні в секундах
SESSION_COOKIE_NAME = 'sessionid'
SESSION_EXPIRE_AT_BROWSER_CLOSE = False
# Налаштування для завантаження файлів
FILE_UPLOAD_MAX_MEMORY_SIZE = 5242880 # 5 MB
FILE_UPLOAD_PERMISSIONS = 0o644
FILE_UPLOAD_DIRECTORY_PERMISSIONS = 0o755
# Налаштування для кешування шаблонів
TEMPLATE_LOADERS = (
    ('django.template.loaders.cached.Loader', (
        'django.template.loaders.filesystem.Loader',
        'django.template.loaders.app_directories.Loader',
    )),
)

```

base.html

```
{% load i18n %}
<!DOCTYPE html>
<html lang="{{ current_language }}">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>{% block title %}{% trans "CMS" %}{% endblock
%}</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bo
otstrap.min.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.0/font/bootstrap-icons.css">
    {% block meta %}{% endblock %}
    {% if settings.GA_TRACKING_ID %}
    <!-- Google Analytics -->
    <script async
src="https://www.googletagmanager.com/gtag/js?id={{
settings.GA_TRACKING_ID }}"></script>
    <script>
        window.dataLayer = window.dataLayer || [];
        function gtag(){dataLayer.push(arguments);}
        gtag('js', new Date());
        gtag('config', '{{ settings.GA_TRACKING_ID }}');
    </script>
    {% endif %}
    <style>
        .hero-section {
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            padding: 100px 0;
            color: white;
            margin-bottom: 2rem;
        }
        .navbar {
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }
        .nav-link.language-switch {
            padding: 0.25rem 0.5rem;
            margin-left: 0.5rem;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-12">
                <div class="hero-section">
                    <h1 class="text-center">{{ title }}
                </div>
                <div class="text-center">
                    <h2>{{ subtitle }}
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

```

        border-radius: 4px;
        background: rgba(255,255,255,0.1);
    }
    .nav-link.language-switch:hover {
        background: rgba(255,255,255,0.2);
    }
    .nav-link.language-switch.active {
        background: rgba(255,255,255,0.3);
    }
    .social-links img {
        transition: transform 0.2s;
    }
    .social-links img:hover {
        transform: scale(1.1);
    }
    .footer {
        background: #2d3436;
        color: #dfe6e9;
        padding: 3rem 0;
        margin-top: 4rem;
    }
    .footer a {
        color: #dfe6e9;
        text-decoration: none;
    }
    .footer a:hover {
        color: #ffffff;
    }
    .footer h5 {
        color: #ffffff;
        margin-bottom: 1.5rem;
    }
    .card {
        border: none;
        box-shadow: 0 2px 15px rgba(0,0,0,0.05);
        transition: transform 0.2s;
    }
    .card:hover {
        transform: translateY(-5px);
    }
</style>
</head>
<body class="d-flex flex-column min-vh-100">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

```

```

<div class="container">
    <a class="navbar-brand" href="{% url 'home' %}">
        <i class="bi bi-grid-3x3-gap-fill me-2"></i>{%
trans "CMS" %}
    </a>
    <button class="navbar-toggler" type="button" data-
bs-toggle="collapse" data-bs-target="#navbarNav">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse"
id="navbarNav">
        {% if menus %}
        {% for menu in menus %}
        <ul class="navbar-nav me-auto">
            {% for item in menu.items.all %}
            {% if not item.parent %}
            {% if item.page %}
            <li class="nav-item">
                <a class="nav-link {% if request.path
== item.page.get_absolute_url %}active{% endif %}"
                    href="{{ item.page.get_absolute_url
}}">
                    {{ item.get_title }}
                </a>
            </li>
            {% else %}
            <li class="nav-item">
                <a class="nav-link {% if request.path
== item.url %}active{% endif %}"
                    href="{{ item.url }}">
                    {{ item.get_title }}
                </a>
            </li>
            {% endif %}
            {% endif %}
            {% endfor %}
        </ul>
        {% endfor %}
        {% endif %}
        <div class="navbar-nav ms-auto">
            <form action="{% url 'set_language' %}"
method="post" class="d-flex">
                {% csrf_token %}

```

```

        <input name="next" type="hidden"
value="{{ request.path }}">
        {% for code, name in languages %}
        <button type="submit" name="language"
value="{{ code }}"
class="nav-link language-
switch {% if code == current_language %}active{% endif %}">
        {{ code|upper }}
        </button>
        {% endfor %}
    </form>
</div>
</div>
</div>
</nav>
{% block hero %}{% endblock %}
<div class="container flex-grow-1">
    {% block content %}{% endblock %}
</div>
<footer class="footer">
    <div class="container">
        <div class="row">
            <div class="col-md-4">
                <h5>
                    <i class="bi bi-grid-3x3-gap-fill me-
2"></i>
                    {% trans "CMS" %}
                </h5>
                <p>{% trans "Система управління контентом"
%}</p>
            </div>
            <div class="col-md-4">
                {% if social_links %}
                <h5>
                    <i class="bi bi-share me-2"></i>
                    {% trans "Соціальні мережі" %}
                </h5>
                <div class="social-links">
                    {% for link in social_links %}
                    <a href="{{ link.url }}"
target="_blank" class="me-3">
                        
                    </a>

```

```

        {% endfor %}
    </div>
    {% endif %}
</div>
<div class="col-md-4 text-md-end">
    <h5>
        <i class="bi bi-info-circle me-2"></i>
        {% trans "Інформація" %}
    </h5>
    <p class="mb-1">© {% now "Y" %} {% trans
"CMS" %}</p>
    <p class="mb-0">{% trans "Всі права
захищено" %}</p>
    </div>
</div>
</div>
</footer>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/boot
strap.bundle.min.js"></script>
    {% block extra_js %}{% endblock %}
</body>
</html>

```

feedback.html

```
{% extends "base.html" %}
{% load i18n %}
{% block title %}{% trans "Зворотній зв'язок" %} - {% trans
"CMS" %}{% endblock %}
{% block content %}
<div class="row">
    <div class="col-md-8 mx-auto">
        <div class="card">
            <div class="card-body">
                <h1 class="card-title">{% trans "Зворотній
зв'язок" %}</h1>
                <form method="post">
                    {% csrf_token %}
                    {% for field in form %}
                        <div class="mb-3">
                            <label class="form-label">{{
field.label }}</label>
                            {{ field }}
                            {% if field.errors %}
                                <div class="alert alert-danger mt-2">
                                    {% for error in field.errors %}
                                        <p class="mb-0">{{ error }}</p>
                                    {% endfor %}
                                </div>
                            {% endif %}
                        </div>
                    {% endfor %}
                    <button type="submit" class="btn btn-
primary">{% trans "Надіслати" %}</button>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

feedback_success.html

```
{% extends "base.html" %}
{% load i18n %}
{% block title %}{% trans "Повідомлення надіслано" %} - {%
trans "CMS" %}{% endblock %}
{% block content %}
<div class="row">
    <div class="col-md-8 mx-auto">
        <div class="alert alert-success">
            <h4>{% trans "Дякуємо!" %}</h4>
            <p>{% trans "Ваше повідомлення успішно надіслано.
Ми зв'яжемося з вами найближчим часом." %}</p>
            <a href="{% url 'home' %}" class="btn btn-
primary">{% trans "На головну" %}</a>
        </div>
    </div>
</div>
{% endblock %}
```

index.html

```
{% extends "base.html" %}
{% load i18n %}
{% block title %}{% trans "Головна" %} - {% trans "CMS" %}{%
endblock %}
{% block hero %}
<section class="hero-section text-center">
    <div class="container">
        <h1 class="display-4 mb-4">{% trans "Ласкаво просимо
до CMS" %}</h1>
        <p class="lead mb-4">{% trans "Сучасна система
управління контентом" %}</p>
        <a href="{% url 'feedback' %}" class="btn btn-light
btn-lg">
            <i class="bi bi-envelope me-2"></i>{% trans
"Зв'яжіться з нами" %}
        </a>
    </div>
</section>
{% endblock %}
{% block content %}
{% if pages %}
<div class="row mb-4">
    <div class="col-12">
        <h2 class="text-center mb-4">
            <i class="bi bi-file-text me-2"></i>
            {% trans "Наші сторінки" %}
        </h2>
    </div>
</div>
<div class="row row-cols-1 row-cols-md-2 row-cols-lg-3 g-4">
    {% for page in pages %}
    <div class="col">
        <div class="card h-100">
            {% if page.featured_image %}
            
            {% endif %}
            <div class="card-body">
                <h3 class="h5 card-title">{{ page.get_title
}}</h3>
                {% if page.meta_description %}
                <p class="card-text text-muted">
```

```

    {{ page.meta_description|truncatewords:20
}}
    </p>
    {% endif %}
    <a href="{{ page.get_absolute_url }}"
class="btn btn-outline-primary">
        <i class="bi bi-arrow-right me-2"></i>{%
trans "Читати далі" %}
    </a>
</div>
<div class="card-footer bg-transparent text-
muted">
    <small>
        <i class="bi bi-clock me-2"></i>
        {{ page.modified|date:"d.m.Y" }}
    </small>
</div>
</div>
</div>
    {% endfor %}
</div>
{% else %}
<div class="text-center py-5">
    <i class="bi bi-journal-x display-1 text-muted mb-3"></i>
    <h2>{% trans "Сторінки відсутні" %}</h2>
    <p class="text-muted">{% trans "На даний момент немає
доступних сторінок." %}</p>
</div>
{% endif %}
{% endblock %}

```

page.html

```
{% extends "base.html" %}
{% load i18n %}
{% block title %}{{ page.get_title }} - {% trans "CMS" %}{%
endblock %}
{% block meta %}
    {% if page.meta_title %}
<meta name="title" content="{{ page.meta_title }}">{% endif %}
    {% if page.meta_description %}
<meta name="description" content="{{ page.meta_description
}}">{% endif %}
    {% if page.meta_keywords %}
<meta name="keywords" content="{{ page.meta_keywords }}">{%
endif %}
{% endblock %}
{% block content %}
<div class="card">
    <div class="card-body">
        <h1 class="card-title">{{ page.get_title }}</h1>
        <div class="card-text">
            {{ page.get_content|safe }}
        </div>
        {% if page.modified %}
        <div class="text-muted mt-3">
            <small>{% trans "Обновлено" %}: {{
page.modified|date:"d.m.Y H:i" }}</small>
        </div>
        {% endif %}
    </div>
</div>
{% endblock %}
```