Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ВЕБ-КОНТЕНТОМ**

Виконав: студент 4 курсу, групи КНТ-527
спеціальності
123 «Комп'ютерна інженерія»
(код і найменування спеціальності)
Освітня програма (спеціалізація)
«Комп'ютерна інженерія»

Юрченко Денис Станіславович
(прізвище та ініціали)
Керівник Касьян М.М.
(прізвище та ініціали)
Рецензент Морщавка С.В.
(прізвище та ініціали)

м. Запоріжжя
2021 рік

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри Кудерметов Р.К
_____ _____
"____" _____ 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Юрченка Дениса Станіславовича
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка системи управління веб-контентом

керівник проекту (роботи) Касьян Микола Миколайович, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "17" березня 2021 року № 81

2. Строк подання студентом проекту (роботи) 10 травня 2021 року

3. Вихідні дані до проекту (роботи) базовий набір функціональних можливостей CMS; зручний інтерфейс панелі адміністрування; забезпечення модульності та можливості розширення системи; простота використання шаблонів; підтримувана мова - PHP

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз технічного завдання; дослідження предметної області та переваг систем управління контентом; розробка структури системи; проектування бази даних системи; розробка програмного ядра системи; тестування розробленої системи управління веб-контентом

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) аналіз предметної області; структура системи управління контентом; структура бази даних системи; інтерфейс системи управління контентом; тестування розробленої системи

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	Касьян М. М., доцент		
Нормоконтроль	Щербак Н.В., ст. викл.		

7. Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	01.03.2021 р.	
2	Дослідження предметної області	15.03.2021 р.	
3	Проектування системи	17.03.2021 р.	
4	Розробка бази даних	20.03.2021 р.	
5	Розробка інтерфейсу системи	25.03.2021 р.	
6	Розробка програмного ядра системи	01.04.2021 р.	
7	Тестування розробленої системи	07.04.2021 р.	
8	Оформлення ПЗ	15.04.2021 р.	
9	Оформлення графічного матеріалу	20.04.2021 р.	
10	Оформлення презентації	01.05.2021 р	

Студент _____ **Д. С. Юрченко**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **М. М. Касьян**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 58 с., 32 рис., 2 лістинги, 13 посилань.

CMS, CSS, JAVASCRIPT, HTML, PHP, ВЕБСАЙТ, ВЕБКОНТЕНТ, ІНТЕРНЕТ, КЛІЄНТ, СЕРВЕР

Об’єктом проектування в дипломній роботі є автоматизована система управління вебконтентом (CMS). Предметом аналізу в дипломній роботі є методи і технології розробки застосунків для вебслужби мережі Інтернет.

Мета проекту – автоматизувати і тим самим спростити розробку вебсайтів для мережі Інтернет.

Для досягнення поставленої мети в роботі потрібно виконати наступні завдання:

- спроектувати програмне ядро системи управління вебконтентом з достатньо високою швидкодією системи;

- спроектувати базу даних для системи управління вебконтентом;

- розробити зручний та простий інтерфейс панелі адміністрування для оперативного управління вебконтентом;

- реалізувати базовий набір функціональних можливостей CMS;

- забезпечити модульність та можливість розширення системи;

- забезпечити безпеку використання та розмежування прав доступу в розрахованому на багато користувачів режимі.

В першому розділі аналізується функціонал існуючих систем управління контентом, технології та програмні засоби розробки вебсайтів та вебзастосунків. В другому розділі описуються етапи розробки системи управління вебконтентом,а в третьому розділі описуються кроки з тестування розробленої системи.

ЗМІСТ

Перелік графічних матеріалів:

Плакат 1 - Принцип роботи та рейтинг популярності CMS систем

Плакат 2 - Схема функціонування CMS

Плакат 3 - Структурна схема бази даних проектованої CMS

Плакат 4 - Тестування функціоналу системи управління

Плакат 5 - Тестування продуктивності розробленої системи в Google Lighthouse

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

CMS (Content management system) - система управління контентом

БД – база даних

СУБД – система управління базою даних

WWW (World Wide Web) – всесвітня комп’ютерна мережа

MVC (Model-View-Controller) – шаблон проектирования (схема разделения данных) Модель-Вид-Контроллер

PHP (Hypertext Preprocessor) - скриптова мова загального призначення що найбільш широко застосовується для розробки вебдодатків

SQL (Structured Query Language) - декларативна мова програмування що застосовується в реляційній базі даних

ВСТУП

Для користувачів комп'ютерів і різних гаджетів мережа Інтернет все сильніше перетворюється з абстрактної всесвітньої комп'ютерної мережі в джерело отримання інформації, що використовується щодня. Все більша кількість людей користується Інтернетом для роботи або відпочинку, пошуку цікавої їм інформації.

З точки зору бізнесу Інтернет це середовище без кордонів, що з легкістю поєднує продавців та покупців, які знаходяться на різних сторонах земної кулі. Нові покупці і ринки збуту - це нові можливості, нові складові доходів. Тому безліч компаній розміщують свої товари і послуги на платформі мережі Інтернет і навіть створюються нові компанії, що спеціалізуються на ведення бізнесу тільки в Інтернеті. У сучасному світі навіть невеликі компанії з регіональним охопленням своєї аудиторії розуміють всі переваги просування своїх товарів і послуг в мережі Інтернет, намагаються впроваджувати нові технології для більшого охоплення місцевого ринку.

На початку появи і розвитку вебтехнологій, вебсайти являли собою досить просту форму представлення даних і складалися в основному з невеликого числа сторінок, пов'язаних гіперпосиланнями. В процесі розвитку мережі Інтернет та зростання її впливу виникла необхідність створення все більш складних з точки зору функціональності і великих з точки зору обсягів інформації вебсайтів. У міру цього, використання старого підходу до створення і особливо управління веб-сайтами ставало все більш складним і трудомістким тому, що управління контентом дуже великої кількості пов'язаних між собою вебсторінок вимагає багато часу і хоча б базових знань мов розмітки та процесів верстки. Для полегшення процесів розробки та управління вебсайтів, а також збільшення набору функціональних можливостей були створені, так звані, системи управління контентом (CMS). Їх головним завданням стало забезпечення та організація спільного процесу створення, редагування і управління вмістом

(контентом) вебсайтів. Особливо це актуально, коли в наведених процесах бере участь велика кількість людей. При цьому достатній рівень їх кваліфікації - впевнений користувач комп'ютера.

За час від появи систем управління вебконтентом з середини 90-х років по сьогодення розроблені сотні різних CMS. При цьому можна виділити такі основні їх функції:

- наявність інструментів для створення контенту, організація спільної роботи над ним;

- управління контентом (зберігання, контроль версій, дотримання режиму доступу, управління потоком документів і т. і.);

- публікація контенту,

- подання інформації в такому вигляді, який був би зручний для навігації і пошуку.

В системі управління контентом можуть зберігатися і оброблятися найрізноманітніші дані - текстові документи, фотографії, відео, аудіо і так далі. Подібні системи часто використовується для зберігання, управління, редагування та публікації документації. При цьому можливість контролю версій є однією з важливих їх переваг, особливо, коли контент редагується великою кількістю людей.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ І ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Класифікація систем управління контентом

Як правило, системи управління контентом діляться на два наступних типи:

- системи управління контентом рівня підприємства (англ. Enterprise Content Management System - ECMS);

- системи управління контентом вебсайту (вебконтентом) (англ. Web Content Management System - WCMS).

У зв'язку з тим, що ECMS мають велику різноманітність, що тісно пов'язана з конкретною предметною областю, в якій вони застосовуються, наприклад, HRM, DMS, CRM, ERP і т. д., поняття CMS істотно частіше використовується стосовно системам управління вебконтентом, відповідно, термін CMS застосовується в основному для позначення WCMS. Такі CMS застосовуються для управління різним контентом вебсайту і надають користувачам простий і зручний інтерфейс та інструментарій для введення, зберігання і публікації інформації, дозволяючи автоматизувати процесирозміщення інформації в базі даних та її вилучення при формуванні вебсторінки мовою гіпертекстової розмітки HTML.

На сьогоднішній день є багато як платних, так і безкоштовних систем управління вебконтентом. За способом роботи їх можна поділити на кілька варіантів:

1) Генерація сторінки на основі отриманого запиту. При такому варіанті система працює за принципом зв'язку модуль редагування - база даних - модуль представлення. Відповідно із запитом, інформація витягується з бази даних і на її основі модуль представлення генерує сторінку з контентом. За допомогою модуля редагування інформація в базі даних може змінюватися. Таким чином, при кожному новому запиті вебсторінка генерується сервером заново. Це створює

додаткове навантаження на ресурси системи. Для зниження такого навантаження в сучасних вебсерверах застосовуються засоби кешування.

2) Генерація сторінки шляхом редагування. Подібні системи орієнтовані на редагування сторінок. В результаті після редагування створюється набір статичних сторінок. При такому варіанті суттєво знижується інтерактивність взаємодії між відвідувачем сайту та контентом сайту.

3) Змішаний варіант поєднує переваги двох попередніх типів і реалізовується на основі кешування. Сторінка один раз генерується за допомогою модуля представлення і потім кешується. Це дозволяє їй завантажуватися з кешу в кілька разів швидше. При цьому оновлення кешу здійснюється або автоматично після закінчення терміну життя, або при зміні певних розділів сайту, або вручну по команді адміністратора. Ще один спосіб - це збереження на етапі редагування сайту деяких інформаційних блоків і подальша збірка вебсторінки з цих блоків, коли користувач запросить відповідну сторінку.

Більшість сучасних CMS володіють модульною архітектурою. Це дозволяє адміністратору довільно вибрати і конфігурувати необхідні йому компоненти.

При цьому, як правило, типовими модулями є:

- динамічне меню,
- новини,
- пошук по сайту,
- опитування,
- блог,
- гостьова книга,
- статистика відвідувань та інші.

Сайти, що розробляються за допомогою систем управління контентом, як правило, використовують такі технології:

- вебсервер;
- сховище даних (переважно СУБД, наприклад, MySQL, PostgreSQL або noSQL);
- візуальний редактор веб-сторінок;

- файловий менеджер з інтерфейсом для управління файлами;
- система управління правами доступу для користувачів і редакторів веб.

1.2 Принцип роботи системи управління вебконтентом

Основний принцип роботи CMS - демонстрація користувачам вебсторінок сформованих "на льоту" (тобто безпосередньо перед відображенням) з попередньо створених шаблонів з певним дизайном і контентом, який зберігається в базі даних.

При такому підході сайту, як набору сторінок не існує. Є тільки окремі шаблони і набір різних матеріалів (текстові файли, зображення, файли з архівами, документи MS Office або PDF і т.д.). Система управління контентом генерує сторінку і відображає її користувачеві в момент запиту. У кожному конкретному випадку користувачеві може бути показана інформація, яка більше ніколи не буде видною, наприклад, вміст кошика товарів в Інтернет-магазині. Це завдання виконує CMS або, так званий, движок сайту. При цьому CMS не тільки генерує сторінки для відображення користувачам, але і намагається максимально прискорити цей процес щоб обробити максимальну кількість запитів в одиницю часу, захищає дані від зловмисників не даючи засмітити базу даних спамом, а також виконує у фоновому режимі багато іншої роботи, необхідної для стабільного функціонування сайту.

Користувачі CMS діляться на дві групи - розробники шаблонів сторінок і розробники контенту (інформаційного наповнення). Одна група користувачів задає структуру і оформлення сторінок, а інша наповнює його змістом.

Функціонал системи управління контентом структурований відповідно до життєвого циклу системи (рис.1.1) [1].

Спочатку група впровадження розгортає ядро CMS і створює сховище контенту в базі даних. Потім адміністратор надає доступ до системи різним

користувачам, далі створюється і публікується контент, до якого застосовуються шаблони оформлення.

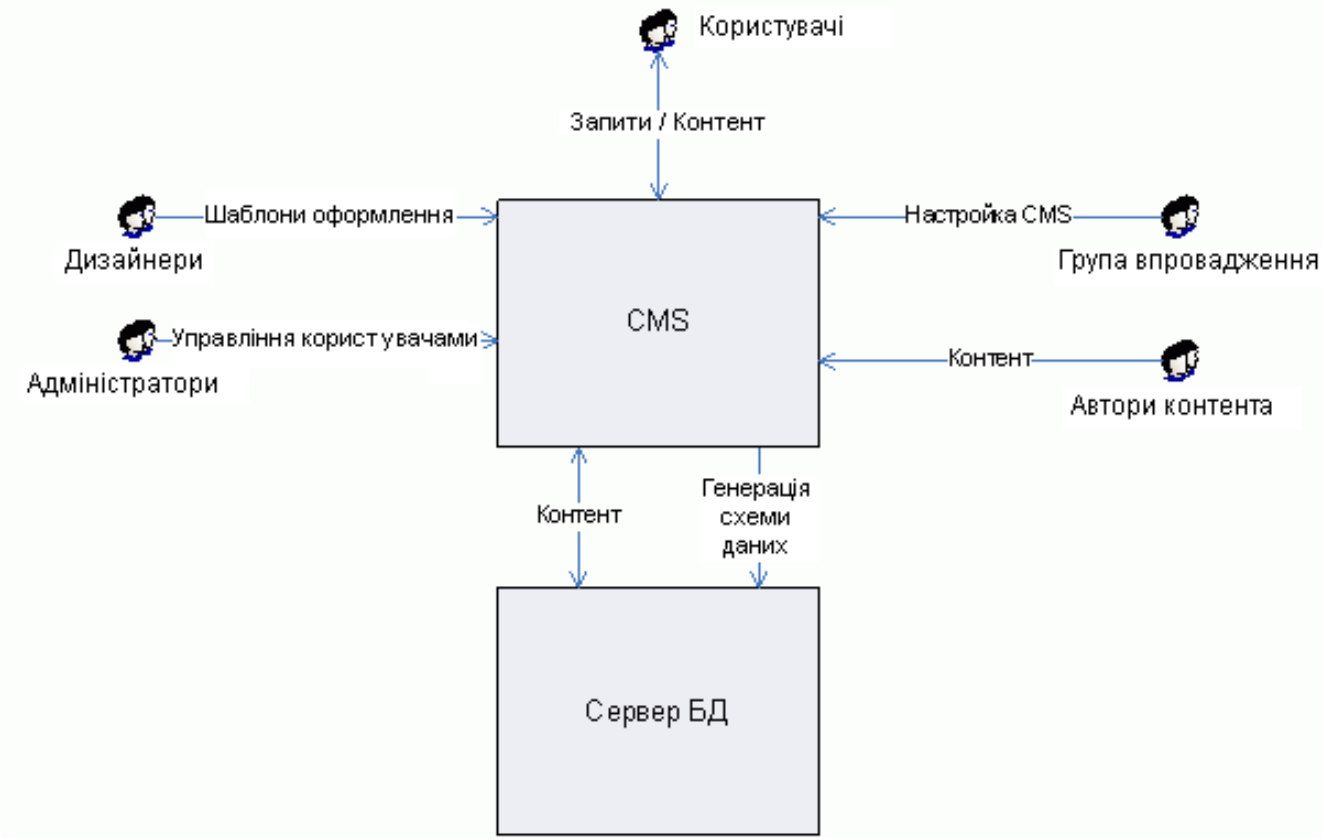


Рисунок 1.1 – Схема функціонування CMS

1.2.1 Створення та редагування контенту

На цьому етапі створюються всі типи контенту, при цьому важлива наявність можливості зберігання інформації про версії контенту. Це дозволяє контролювати операції зміни контенту і при необхідності його відновити. Також можна в будь-який момент скасувати зміни і відкотитися на одну з попередніх версій. Крім того контроль версій потрібен для визначення відповідальних осіб, а також резервного відновлення системи.

Коли всі типи контенту створені, його автори отримують можливість створювати, редагувати і видаляти елементи контенту обраного типу. Сама CMS також містить певний набір візуальних компонентів, наприклад, для редагування тексту, вибору зображень, вибору дизайну відображення і т. д.

Крім можливості редагування елементів контенту, необхідно передбачити можливість розбиття контенту на категорії або рубрики.

1.2.2 Створення шаблонів оформлення

Для вирішення проблеми представлення контенту в системах управління вебконтентом використовується технологія шаблонів, які і визначають зовнішній вигляд сторінки. При цьому розробнику шаблонів не обов'язково знати всі технічні аспекти. На ранніх етапах існування WWW шаблони представляли собоюзаготовки HTML-коду, з якого шляхом маніпуляцій в HTML-редакторі виходили готові вебсторінки. На сучасному етапі такими заготовками маніпулюють вже не дизайнери в своїх редакторах, а серверні вебзастосунки. Відповідно, сучасний шаблон вебсторінки це блок HTML-коду, який за допомогою спеціальних тегів або впроваджених сценаріїв, спрощує включення динамічна згенерованого контенту на етапі виконання. Для застосування подібних шаблонів програмістам необхідний певний стандартизований інтерфейс або, так званий, шаблонний движок (від англійського template engine), який також може включати в себе різні додаткові функції, наприклад, підтримку кешування шаблонів, їх динамічне оновлення і т. д.

1.2.3 Публікація контенту

Механізм публікації інформації в системі управління контентом має на увазі як процес створення, редагування та видалення шаблонів вебсторінок, так і відповідність типів контенту і шаблонів сторінок. До складу додаткових можливостей системи з точки зору публікації також може входити попередня генерація статичної версії сайту (може бути корисно при розміщенні інформаційної системи на апаратній платформі з обмеженими можливостями).

Типовий процес публікації інформації в WWW виглядає наступним чином. Звичайним прийомом форматування і оформлення інформаційного наповнення є шаблони подання контенту, тому першим етапом цього процесу є створення наборів шаблонів. Типовий шаблон містить HTML-розмітку і місця, куди в подальшому будуть вставлятися дані. Потім на основі цих шаблонів автори інформаційного наповнення створюють сторінки і подають їх редакторам для

затвердження. Редактори, відповідно, можуть або відхилити сторінку і повернути її автору на доопрацювання, або затвердити її і передати модератору сайту. При першому варіанті запускається нова ітерація процесу, при другому модератор сайту звіряє розташування сторінки на сайті, дату і термін її публікації і, якщо все добре, сторінка стає видною користувачам.

1.2.4 Керування користувачами

Управління користувачами включає створення, редагування і видалення облікових записів користувачів і груп користувачів, а також надання прав доступу для роботи з конкретними елементами контенту. Важливою умовою є наявність профілів (profiles), за допомогою яких можна згенерувати персональне представлення контенту для конкретного користувача. Також корисною буде надання користувачам можливості делегувати свої права. Це, наприклад, дозволить перепризначати відповідального виконавця конкретної роботи і виключити простої через відсутність певної людини.

Подання інформації створюється на основі даних, а також вподобань певного користувача. Персоніфікація досягається шляхом використання профілів, тобто спеціальних записів, в яких зберігається інформація, оптимізована під конкретних користувачів системи.

1.3 Рейтинг популярності CMS систем

На даний час є багато різних системи управління вебконтентом - платні і безкоштовні, побудовані на базі різних технологій (найчастіше PHP), а також універсальні і спеціалізовані.

Більшість з них представляють собою візуальний редактор, в якому користувач при бажанні має можливість збирати сайт, як конструктор з наявних блоків, не вдаючись у тонкощі програмування і розмітки.

Одні системи є спеціалізованими і націлені на вирішення конкретних завдань (Інтернет-магазини, форуми, блоги), інші є універсальними і зручні при реалізації найрізноманітнішого функціоналу сайту. Деякі CMS складаються з різноманітних функціональних блоків або модулів, інші монолітні і не допускають довільної конфігурації.

Частина систем безкоштовні і дозволяють користувачам самим вносити в них зміни, інші продаються за гроші і не надають можливості редагування програмного ядра системи.

Рейтинг популярності CMS систем на поточний момент завжди можна простежити на спеціалізованих вебресурсах таких, як w3techs.com або itrack.ru. Наприклад, за даними ресурсу itrack.ru [2] за березень 2021 року рейтинг популярності безкоштовних CMS виглядає наступним чином (рис. 1.2). Розглянемо першу трійку рейтингу докладніше

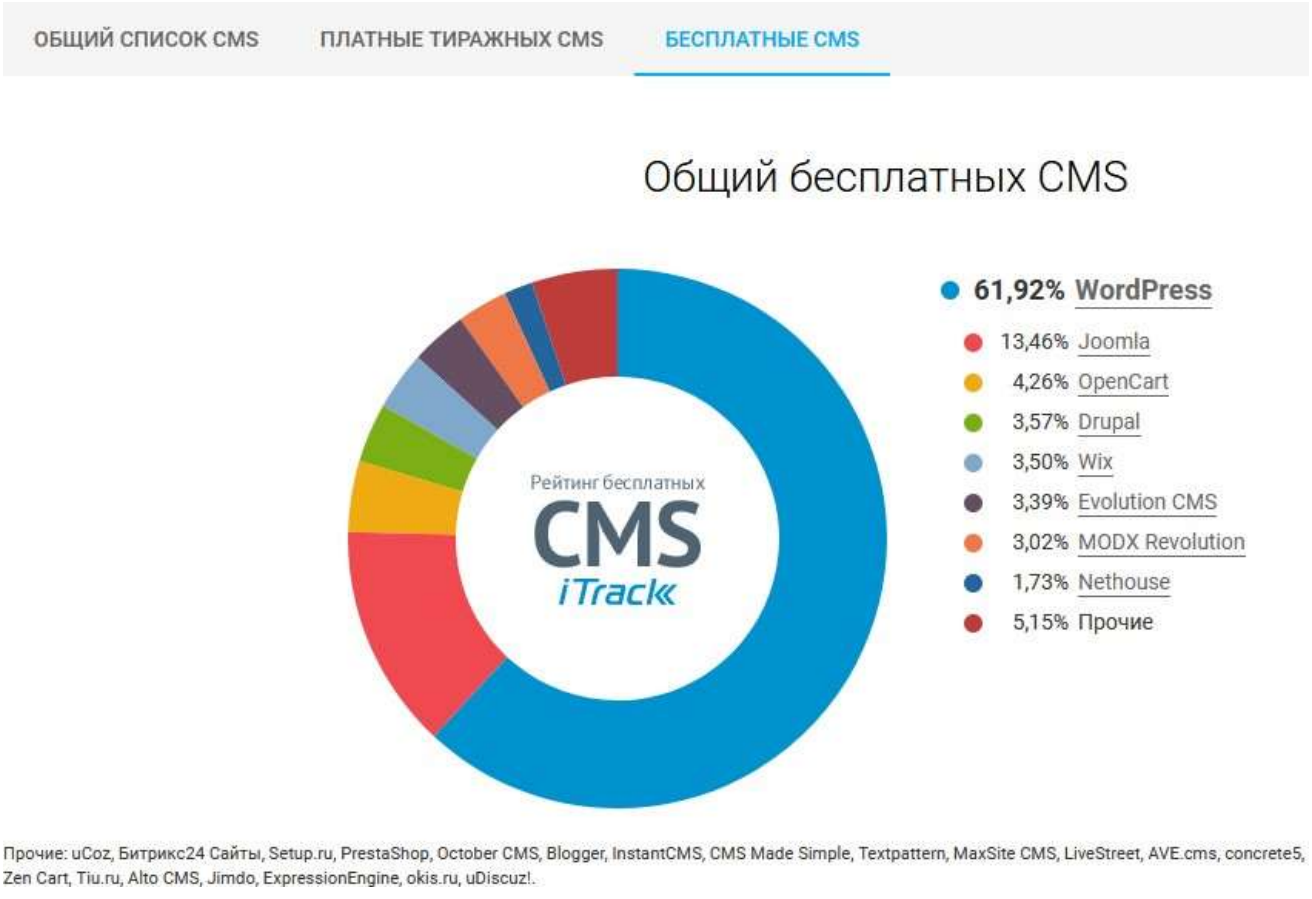


Рисунок 1.2 – Рейтинг популярності безкоштовних CMS

1.3.1 Система WordPress

Вже досить давно система WordPress є найпопулярнішою системою управління контентом, більшість найбільш поширених Інтернет-ресурсів створюються і супроводжуються за допомогою саме цієї CMS. Ця CMS знаходиться у вільному доступі, вона зручна і проста в установці та налаштування, до того ж має підтримку HTML і CSS, досить велика кількість плагінів і зручний інтерфейс для роботи з ними і дуже багато шаблонів у вільномудоступі.

До недоліків цієї CMS можна віднести велику кількість вразливостей і багів, тому сайти на цій системі можуть бути схильні до злому. Крім того так, як процес створення сайту на WordPress полягає в використанні набору готовихплагінів, це значно обмежує можливість реалізації якогось нестандартного функціонала. Структура самого програмного движка цієї CMS досить складна, а документація від розробників далеко не повна. До того ж ця система не дозволяє розробляти високонавантажені проекти.

1.3.2 Система Joomla

З одного боку ця сучасна CMS постійно оновлюється, досить зручна в розробці і використанні система, до того ж є безкоштовною. З іншого боку, їй властивий ряд недоліків, наприклад, можливі труднощі при оновленні версії CMS, що може привести до порушення деяких вже встановлених компонентів і дозволить провести лише покрокове оновлення через всі проміжні версії.

Крім того Joomla в числі систем, які найбільш часто піддаються атакам зловмисників і зломам. Головною причиною цього є не дуже висока захищеність системи, хоча поступово ситуація поліпшується завдяки роботі спільноти Joomla.

Ще один недолік, це складність в освоєнні системи. Вона не настільки інтуїтивно зрозуміла і, якщо раніше розробник, наприклад, використовував тільки чистий PHP, то швидко вникнути в процес роботи CMS не вийде.

1.3.3 Система Open Cart

OpenCart - це CMS, яка найкраще підходить для Інтернет-магазину. Вона може бути встановлена на будь-якому вебсервері, який підтримує зв'язку PHP і

MySQL. Система також поширюється безкоштовно, побудована на основі шаблону проектування MVC, включає досить багатий функціонал, необхіднийдля Інтернет-магазину, досить проста у використанні. Структури програмного движка і бази даних системи досить прості, що дозволяє без зайвих складнощів дописувати програмний код і розширювати функціонал. Але сам движок досить обмежений в можливостях, відповідно, система не дозволяє створювати вебсайти різної спрямованості, і є скоріше спеціалізованої.

1.4 Обмеження та недоліки існуючих CMS систем

Завдяки чіткому механізму розширення функціоналу сайтів, що створюються за допомогою CMS, у них дуже низький поріг входу, що дозволяє розробляти сайти навіть людям, далеким від програмування.

Однак, у міру нарощування функціоналу, до програмування все-таки доведеться повернутися. І в цьому плані у CMS є ряд певних недоліків, які пов'язані з особливостями написання коду для обраної конкретної платформи.

1) Необхідність розбиратися зі структурою і особливостями програмного коду щоразу, коли доводиться освоювати нову CMS. Причому, іноді цей процес може бути досить довгим через складність коду або механізмів встановлення розширень.

Якщо з точки зору користувача, який використовує CMS як конструктор сайту лише з установкою базових модулів і шаблонів, різні CMS може бути і схожі, то для розробника-програміста, який має справу з кодом ядра системи, це зовсім не так. Структура коду в різних CMS істотно відрізняється. Навіть якщо, наприклад, CMS характеризується як система на базі MVC, це зовсім не означає, що з нею буде легко і просто розібратися навіть, якщо у розробника вже є досвід роботи з подібними системами.

Наприклад, і OpenCart і Magento позиціонуються, як системи на базі MVC. Але в OpenCart використовується класична MVC модель коду з окремими каталогами для моделей, контролерів і видів (шаблонів), а в Magento всі моделі і контролери розкладені по папках модулів, в залежності від їх функціоналу.

2) Обмеженість функціоналу CMS змушує розробника писати код самостійно для будь-якої нестандартної задачі, що може зажадати багато часу.

Ця ж необхідність властива і для фреймворків, але CMS не такі гнучкі, як фреймворки і при необхідності створення чогось унікального на базі CMS, доведеться постійно упиратися в обмеження коду, встановленими CMS. Тому тривалість розробки може значно збільшуватися в порівнянні з фреймворком.

3) Хоча процес створення сайту за допомогою CMS є більш швидким порівняно зі створенням сайту за допомогою фреймворка, за цю швидкість доводиться доплачувати щоб придбати або коробковий варіант CMS, абододаткові модулі і шаблони для безкоштовних CMS.

Якщо ж брати безкоштовну CMS і самому створювати для неї аналогічні платним модулям доповнення, то в даному випадку вже не доводиться говорити про швидкість розробки.

4) З шаблонами, модулями та іншими розширеннями пов'язаний ще один мінус використання CMS. Причина в тому, що, не дивлячись на їх велику різноманітність, якість коду в них залишає бажати кращого.

Це особливо стосується безкоштовних розширень, які пишуть всі, кому не лінь. Хоча і для платних розширень гарантію сто процентної працездатності ніхто не дасть.

Крім того досить часто різні розширення можуть конфліктувати один з одним, а також з написаним кодом інших програмістів. Все це дуже ускладнює і подовжує за часом процес розробки.

Тому на фріланс-біржах навіть існує окрема категорія замовлень, яка пов'язана з усуненням неполадок, викликаних розширеннями, а також усунення конфліктів між ними.

Це все призводить до істотного зростання часу витраченого на розробку, який часто непропорційний оплаті таких замовлень.

5) Завдяки тому, що CMS є готовим сайтом, вона включає багато можливостей, які можуть бути затребувані розробниками в процесі створення сайтів на її базі. При цьому далеко не завжди розробникам необхідні всі функції, які йдуть, що називається, «з коробки».

В результаті може виникнути ситуація, коли більша частина можливостей може виявитися незатребуваною, одночасно перевантажуючи сайт і сповільнюючи його роботу. Тому в деяких випадках відключення непотрібних функцій CMS може зажадати більше часу і сил навіть, ніж створення цього ж сайту за допомогою фреймворка.

Це особливо актуально для CMS, які спочатку створювалися для розробки на їх основі високонавантажених проектів рівня підприємства, наприклад,Magento або 1С-Бітрікс. Вони з самого початку розроблялися для великих масштабів, тому і базовий функціонал у них відповідний.

6) Ще один недолік CMS з точки зору розробника пов'язаний з великою популярністю цих систем і, як наслідок, величезною конкуренцією на ринку праці серед розробників. Тому дуже складно отримати замовлення і на початковому етапі доводиться працювати за невелику плату, щоб набрати на біржі рейтинг і надалі мати перевагу серед інших розробників при подачі заявки на виконання замовлення. Це особливо актуально для зарубіжних бірж, де розробників на порядок більше.

Тепер розглянемо недоліки CMS з точки зору замовника.

1) Якщо розширити функціонал за допомогою сторонніх розширень можна і без допомоги програміста, то усунути проблеми в роботі сайту, пов'язані з їх конфліктами між собою, без допомоги програміста вже не вийде.

При цьому важливо робити бекапи перед кожною установкою будь-яких доповнень, щоб можна було відкотитися назад тому, що невідомо, як зміниться робота сайту після їх встановленні. При наявності ж резервної копії завжди можна

швидко відновити працездатність сайту і бути впевненим, що проблема викликана конкретним доповненням.

2) При бажанні реалізувати якийсь унікальний функціонал або дизайн, якого не можна досягти встановленням стандартних розширень, потрібно бути готовим до додаткових фінансових витрат так, як правило, в рамках обраної конкретної CMS, це зробити може бути складно і, відповідно, дорого. Тим більше що такі дії зможе здійснити тільки досвідчений розробник добре знайомий з CMS.

3) Велика кількість додаткових розширень для CMS може бути як плюсом, так і мінусом тому, що якість їх написання може не завжди бути хорошою.

Наприклад, це може привести до зниження продуктивності сайту, проблемам з його безпекою та конфліктами з іншими розширеннями.

З урахуванням всіх перерахованих вище обмежень і недоліків CMS і, незважаючи на безліч переваг від їх використання, не можна розглядати цейваріант як ідеальне рішення для розробки сайтів.

1.5 Технології розробки вебсайтів та вебзастосунків

Безумовно базовими вебтехнологіями є мова розмітки HTML і мова стилів CSS, які використовують майже всі вебсторінки в мережі Інтернет.

1.5.1 Мови HTML і CSS

HTML5 на сьогодні є стандартом серед мов розмітки в Інтернеті. Мова розмітки є засобом створення текстового документа зі спеціальним синтаксисом, що визначає, як він повинен бути інтерпретований та виведений на екран браузером. HTML-розмітка описує структуру вебсторінки за допомогою тегів, які повідомляють браузеру, як представити всі елементи сторінки, такі як, наприклад, заголовки, текст, посилання та інші, що включені в документ. Тобто задають зміст вебсторінки.

CSS (каскадні таблиці стилів) це мова стилів, що надає розробникам більше ніж HTML можливостей з форматування та оформлення вебсторінки. Наприклад, більш широкий контроль за такими речами, як кольори, шрифти, позиціонування і загальний дизайн вебсторінки. Розділяючи форму відображення (CSS) від змісту (HTML) вебдокументи зручніше як розробляти, так і редагувати в процесі супроводу.

Використання разом цих двох мов достатньо, щоб забезпечити структуру і стиль для статичного вебсайту, тобто такого, де вебсторінки мають фіксований контент, а гіперпосилання є єдиною формою інтерактивності. Клієнтські і серверні мови програмування надають цілу низку додаткових можливостей, що дозволяють розробляти динамічні та більш інтерактивні вебсайти, які можутьадаптуватись під кожного відвідувача індивідуально.

1.5.2 Мова JavaScript

В сьогодні мова сценаріїв JavaScript є фактично стандартом для створення програмного забезпечення вебсайтів та вебзастосунків, що працює на боці клієнта. Тобто разом з HTML-кодом сценарії JavaScript завантажуються через мережу на комп’ютер клієнта та обробляються його браузером. Це дозволяє додати вебсторінці бажаної динаміки та інтерактивності, та зробити реакцію на будь-які дії користувача практично миттєвою. Разом з HTML та CSS, JavaScript є базовою технологією, на якою будується інтерфейс сучасного динамічного та інтерактивного вебсайту. Навіть завдяки тому, що JavaScript працює на стороні клієнта, деякі сайти можуть залишатися інтерактивними навіть при відсутності інтернет-з'єднання.

Інструментальні програмні засоби JavaScript досить розвинені та охоплюють дуже широке коло бібліотек та фреймворків, що значно спрощують роботу розробника. Це, наприклад, такі засоби автоматизації, як Angular, React, Ember, Vue, jQuery, Backbone та інші.

В останні роки мова JavaScript набуває популярності і в якості мови розробки серверної частини вебсайтів та веб-вузлів. Сценарій на боці сервера як правило потрібний, коли актуальним є, наприклад, взаємодія сайту з базою даних.

1.5.3 Серверний JavaScript

Програми, написані на JavaScript, можуть виконуватися на серверах, що використовують Java 6 і пізніших версій. Ця обставина використовується для побудови серверних додатків, що дозволяють обробляти JavaScript на стороні сервера.

Серверний JavaScript та ж сама мова JavaScript, але він набув величезної популярність саме як серверна мова, коли платформа Node.js реалізувала комбінування движка Google V8 JavaScript, циклу подій і низькорівневого API вводу-виводу. Результатом став гнучкий, простий в управлінні високопродуктивний вебсервер, який можна використовувати для розробки серверних застосунків будь-кому, хто вже знає мову JavaScript.

1.5.4 Мова Java

Розроблена компанією Sun Microsystems ще в 90-х роках мова Java має величезне значення як мова програмування загального призначення. Сьогодні Java дуже відома як мова для створення застосунків для Android та десктоп систем, але вона також має довгу історію і як серверна технологія. Java - сервлети та JSP (серверні сторінки Java) є прикладами серверних рішень, що використовують мову Java.

Походячи з мови програмування C, мова Java дуже часто демонструє високі показники при тестуванні продуктивності. Завдяки цьому вебсайти з дуже високим навантаженням, наприклад, Alibaba, Linkedin, Chase та інші, використовують Java. Також причина може бути в тому, що Java - це достатньо зріла мова програмування, що має довгу історію використання у всьому спектрі розробки програмного забезпечення. Існує дуже багато документації, бібліотек і фреймворків, призначених для розробки застосунків промислового рівня, включаючи безпеку, електронну комерцію та інші.

1.5.5 Мова Python

Мова програмування загального призначення Python є мовою з відкритим вихідним кодом, що робить наголос на легкому для читання коді. Python має велику стандартну бібліотеку, що наповнена попередньо закодованими функціями

для кожного випадку, що дозволяє програмістам діяти продуктивніше з меншою кількістю рядків коду. Заснований на мові Python фреймворк Django характеризується швидким прототипуванням і розробкою, що йому перевагу при розробці стартапів, наприклад, таких як Pinterest та Instagram.

1.5.6 Мова Ruby

Мова Ruby схожа на Python тим, що вона також є інтерпретованою мовою, що робить наголос на продуктивності завдяки короткому та простому синтаксису. Але є і відмінність в тому, що коли Python дотримується правила одного вірного способу програмування, Ruby підтримує гнучкість, бо дозволяє декілька способів зробити одне й теж, хоча деякі способи можуть бути швидшими за інші. Тому обрана мова скоріше є питанням вподобання.

Спеціалізований фреймворк Ruby on Rails має відкритий вихідний код, що робить Ruby потужним та актуальним засобом серверної розробки. Цей фреймворк з архітектурою MVC, що дозволяє розробникам швидко завантажувати проекти засновані на перевірених кращих практиках. Такі відомі сайти як, наприклад, Hulu, Basecamp, Shopify, Groupon та інші використовують Ruby.

1.5.7 Мова C#

Мова C#, що була відповіддю Microsoft на мову Java, це мова програмування заснована на мовах C та C ++. Вона, відповідно, використовується для розробки програмного забезпечення для платформи .NET від Microsoft. Тобто,якщо задачею є створення вебсайтів та вебзастосунків для екосистеми Microsoft, C# то є найкращий вибір. Такі вебсайти як, наприклад, MSN, Salesforce та, звичайно, власний вебсайт компанії Microsoft, це приклади сайтів, що використовують C # та платформу ASP.NET.

1.5.8 Мова PHP

Згідно зі звітом W3Tech [3] від квітня 2021 року (рис.1.3), PHP використовує 79,2% вебсайтів як частину їх back-end технологій. PHP - це вбудована в HTML мова сценаріїв, що ідеально підходить для веб шаблонів та систем управління контентом. Найбільш популярні CMS такі як WordPress або

Joomla використовують саме PHP. Мова PHP є перевіреною часом серверної технологією і задіяна в back-end компонентах таких сайтів, як Facebook, Baidu та Wikipedia.

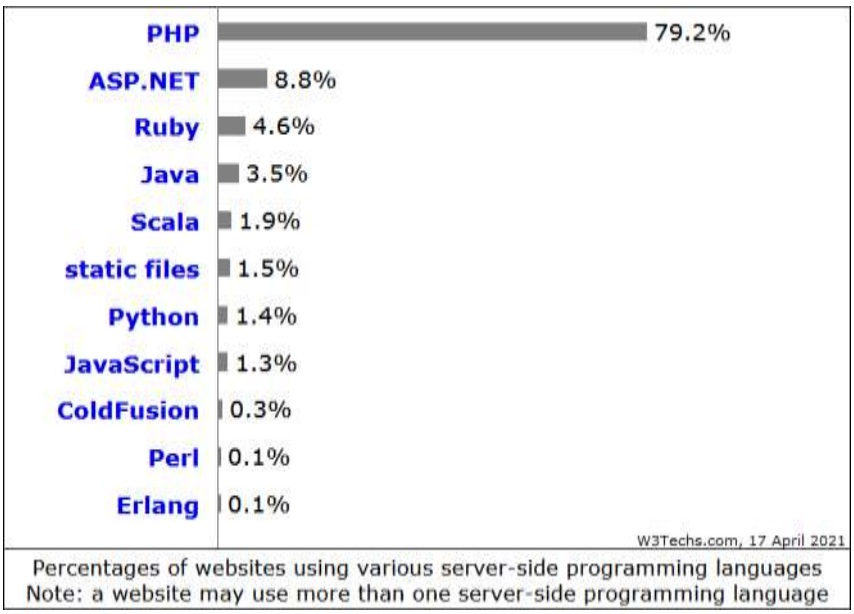


Рисунок 1.3 – Пулярні серверні мови програмування

1.5.9 Вибір середовища розробки

Не менш важливим фактором є наявність зручного редактора вебсторінок для створення і редагування HTML, CSS та JavaScript коду. Notepad (Windows) і TextEdit (MAC) - гарні інструменти для початку, але з накопиченням досвіду роботи хочеться використовувати більш зручний інструмент.

Редактори веб-сторінок можна розділити на дві групи:

- візуальні редактори дозволяють будувати макет сторінки і задавати стилі візуально аналогічно, як в текстовому редакторі MS Word. Вони є зручним інструментом для побудови дизайну сторінки, але, як відомо кожному досвідченому веб-дизайнеру, згенерований редактором код все одно доведеться редагувати для усунення його надмірності.

- текстові редактори, що використовуються для безпосереднього редагування HTML та CSS-коду. Деякі редактори є загальними і не мають спеціальних опцій щодо підтримки веб-коду. Інші редактори є спеціалізованими

для використання веб-мов HTML, CSS, JavaScript та PHP і мають вбудовані засоби для швидкого введу HTML-тегів, CSS властивостей, і тощо.

Після аналізу великої кількості вебредакторів для виконання даного проекту було обрано редактор PHP Storm, що підтримує синтаксис багатьох мов, підтримує пряме підключення до вебсервера, контролює зв'язок між файламикоду, класами і методами, підсвічує код і має в своєму складі відладчик.

Цей редактор надає розробнику можливість управління файлами, що знаходяться на сервері, це дозволяє без використання додаткових програмних засобів вносити зміни в код і відразу перевіряти результат виконання. Дуже зручний у використанні та має інтуїтивно зрозумілий інтерфейс.

1.5.10 Вибір шаблону проектування

Шаблоном проектування (Design Pattern) називають підхід до рішення проблем, що часто зустрічаються при розробці програмного забезпечення. Шаблон проектування не є готовим рішеннями, що можна безпосередньо перетворити в код, а надає загальний опис вирішення проблеми, що можна використовувати в різних ситуаціях.

Найбільш розповсюдженим є шаблон проектування Model-View-Controller MVC (модель-вид-контролер). Цей шаблон є схемою застосування декількох шаблонів проектування, за допомогою яких модель застосунку, інтерфейс користувача та взаємодія з користувачем поділені на три окремих компоненти таким чином, щоб модифікація одного з компонентів надавала мінімальний вплив на інші компоненти. Така схема проектування доволі часто застосовується для побудови архітектурного каркасу, при переході від теорії до реалізації вконкретній предметній області.

Головна мета використання такої концепції полягає у відокремлені бізнес-логіки (моделі) від її візуалізації (виду). Завдяки такому поділу підвищується можливість повторного використання. Найбільш ефективне застосування цієї концепції має місце в тому разі, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах або з різних точок зору.

Завдяки цьому, ефективно вирішуються наступні завдання:

- до однієї моделі можна застосувати декілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, ті самі дані можуть бути одночасно представлені у виді електронної таблиці, гістограми або кругової діаграми;

- не чіпаючи реалізації видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних, тощо) - для цього досить застосувати лише інший контролер;

- деякі розробники спеціалізується лише в однієї області - або розроблюють графічний інтерфейс, або розроблюють бізнес-логіку. Тому можливо таке, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі навіть не будуть знати, який вид буде використовуватись.

1.6 Висновки за розділом

Система управління контентом (CMS) складається з шаблонів та програмних модулів, за допомогою яких автоматизується робота вебсайту.

Основні фактори, що обумовлюють доцільності розробки власної системи управління вебконтентом наступні:

- легкість. У сайті зробленому на самостійно написаному програмному движку буде лише той функціонал, що потрібний, тому такий сайт буде швидше працювати та займати менше місця на хостингу;

- функціональність. При створенні сайту на самостійно написаному програмному движку немає ніяких обмежень в функціоналі. Якщо потрібно зробити щось особливе, програмісту буде простіше це зробити;

- ціна. Розробникам, що використовують самописну CMS не доводиться додавати вартість CMS до ціни сайту;

- гнучкість. У самостійно написаному програмному коді набагато простіше відредагувати якийсь модуль або додати новий;

- досвід. Під час створення власної CMS набувається цінний власний досвід у веброзробці.

При розробці власної системи управління в якості зразку використовувались найпоширеніші CMS WordPress та Joomla. Як основа було обрано той функціонал, що найчастіше використовується саме в цих системах.

2 РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ВЕБКОНТЕНТОМ

2.1 Архітектура системи управління вебконтентом

В загальному вигляді архітектуру системи управління вебконтентом можна представити наступним чином (рис.2.1).

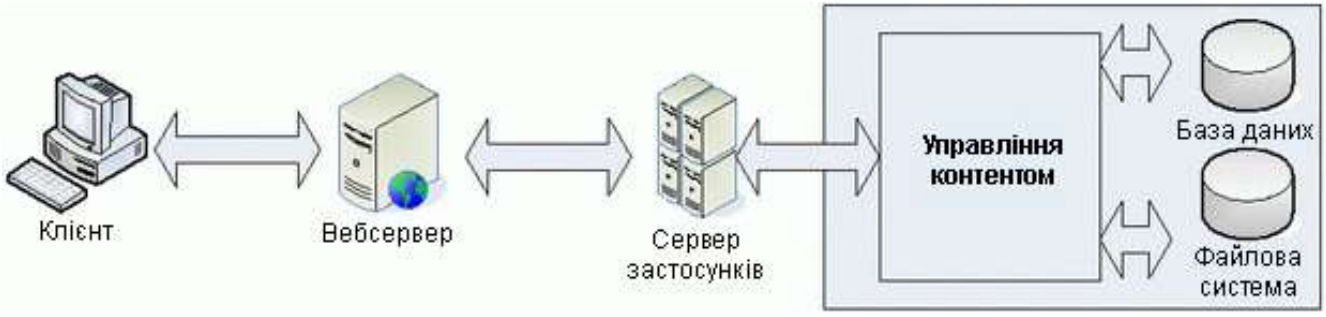


Рисунок 2.1 - Архітектура CMS

В основі цієї технології лежить архітектура клієнт/сервер, що складається із трьох ланок. Така архітектура розподіляє процес обробки даних між клієнтом, сервером застосунків і сховищем даних. На відміну від традиційної двохланкової архітектури в неї є сервер застосунків як проміжна ланка між клієнтом та сховищем даних.

Крім того в системі присутні два сховища. В першому (це, зазвичай, реляційна СУБД) зберігаються всі дані, що публікуються на сайті. В другому (зазвичай, це файлова система) зберігаються елементи представлення - шаблони, графічні зображення і т. д.

Отримуючи запит, сервер застосунків обробляє його, зв'язуючись зі сховищем даних, в якому б місці необхідні дані не знаходилися. Клієнт лише отримує результат у вигляді HTML-файлу. Таким чином, сервер застосунків є стандартизованої платформою для динамічної доставки контенту і побудови основних застосунків. Серверів застосунків може бути й багато, але зв'язок з ними відбувається через вебсервер.

Класична схема побудови багатьох CMS-систем заснована на відокремлені візуального дизайну сайту від його інформаційного наповнення. При створення сайтів за допомогою такої системи розробляється набір шаблонів вебсторінок, в яких згодом розміщується контент, а також бази даних для його зберігання (рис.2.2).

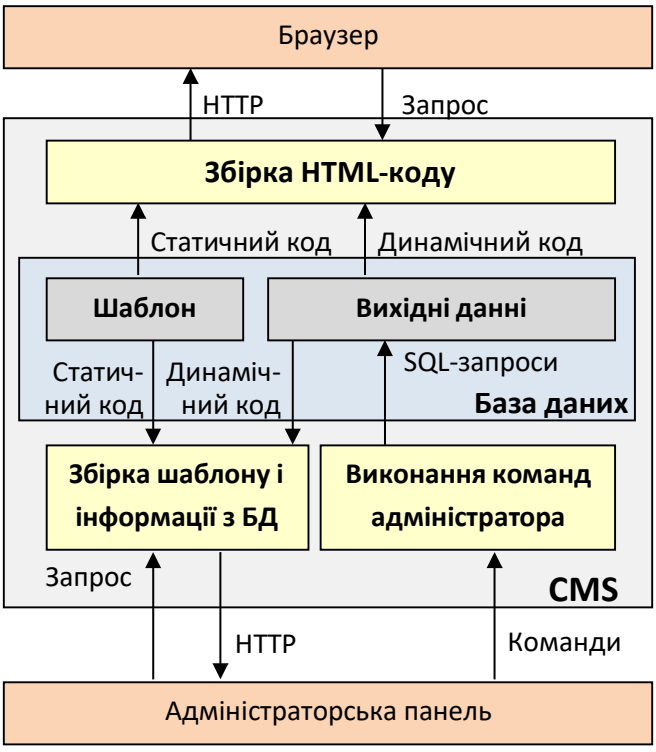


Рисунок 2.2 - Схема функціонування CMS

За результатами аналізу існуючих аналогічних систем, для побудови структури розроблюваної в дипломній роботі системи управління вмістом, було

прийнято рішення використовувати шаблон проектування MVC, що є найбільш зручним та поширеним при розробці подібних автоматизованих систем (рис.2.3).

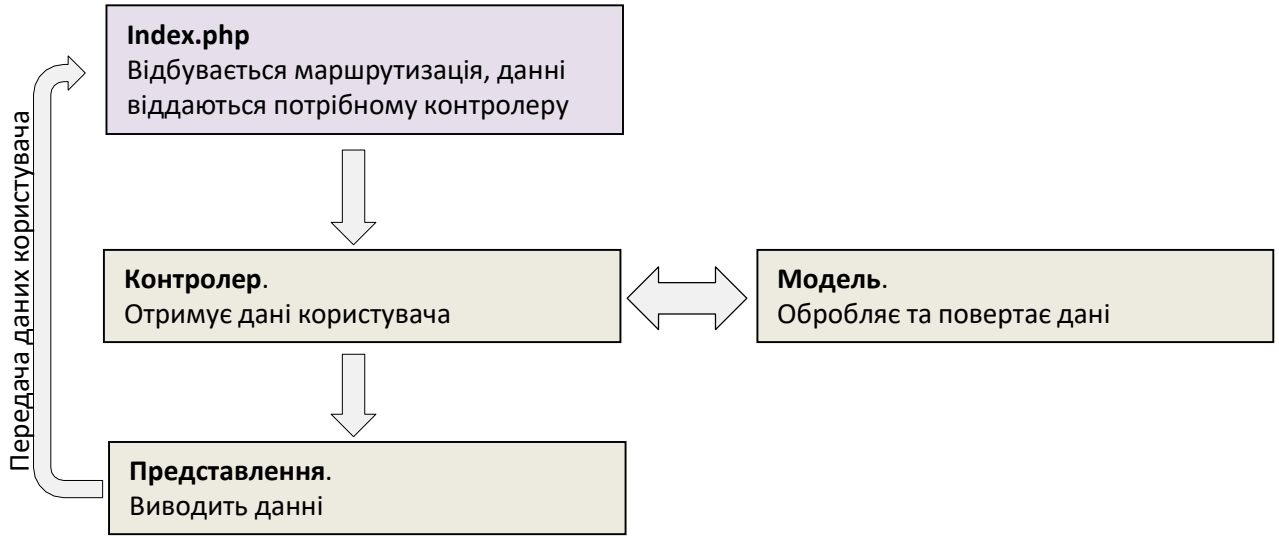


Рисунок 2.3 - Схема функціонування ядра CMS
побудованої на базі шаблона MVC

Таким чином, при розробці власної CMS необхідно вирішити наступні основні завдання:

- спроектувати програмне ядра застосунку;
- спроектувати бази даних системи;
- спроектувати зручний та простий інтерфейс панелі адміністрування для оперативного управління контентом;
- реалізувати базовий набір функціональних можливостей CMS;
- забезпечити модульність та розширюваність системи;
- забезпечити безпеку використання та розподілення прав доступу в багато-користувальницькому режимі;
- можливість використання шаблонів різного типу та складності.

2.2 Розробка бази даних системи управляння вебконтентом

За результатами аналізу вихідних вимог до проектованої системи управління вебконтентом, була спроектована база даних системи, структура якої має наступний вигляд (рис.2.4).

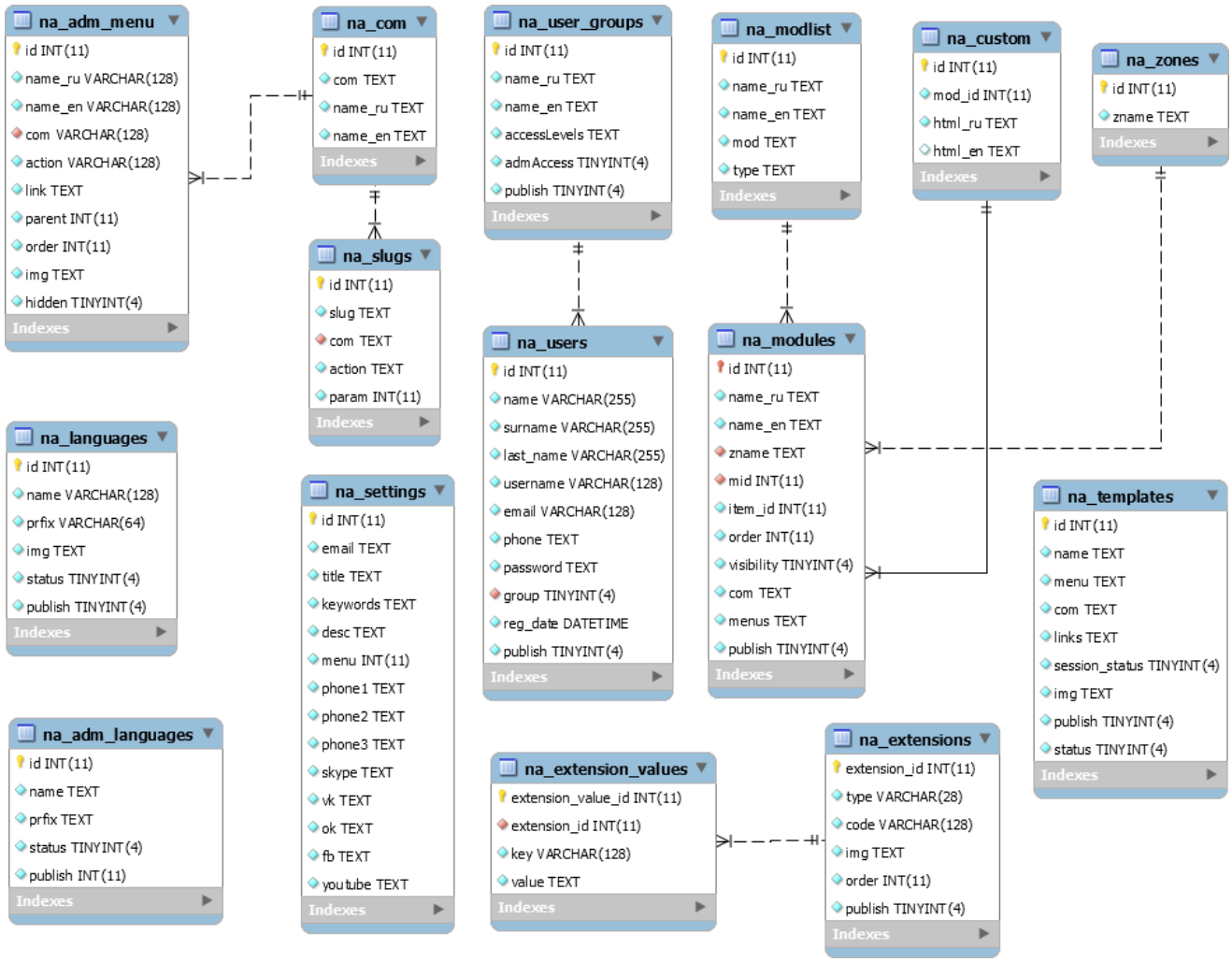


Рисунок 2.4 - Структурна схема бази даних проектованої CMS

База даних призначена для зберігання переліку компонентів та модулів з їх прив'язкою до генерованих вебсторінок. А також даних користувача, груп та прав доступу до набору дій і компонентів в панелі адміністрування.

Таким чином, основні дані для зберігання в БД наступні:

1) Список шаблонів, що використовуються. Для підтримки можливості використання системою багатьох шаблонів необхідно зберігати набір даних, що визначатиме список шаблонів та порядок їх включення.

По кожному з шаблонів повинні зберігатися наступні дані:

- ідентифікатор запису;
- назва шаблону;
- JSON набір пунктів меню до яких прив'язаний шаблон;
- JSON набір імен компонентів до яких прив'язаний шаблон;
- JSON набір зовнішніх посилань до яких прив'язаний шаблон;
- статус сесії при якому шаблон буде використовуватися;
- зображення шаблону;
- стан (публікація) шаблону;
- прапорець визначення основного шаблону (рис.2.5).

Рисунок 2.5 – Структура таблиці шаблонів

2) Основний функціонал проектів, що розробляються складають компоненти. Це набір представлень та відповідних до них оброблювачів, що поєднує в собі функціонал кожної окремої гілки. Для реалізації багато-користувальницького режиму та розподілення прав доступу до будь-яких дій в панелі адміністрування необхідний набір відповідних даних (рис.2.6):

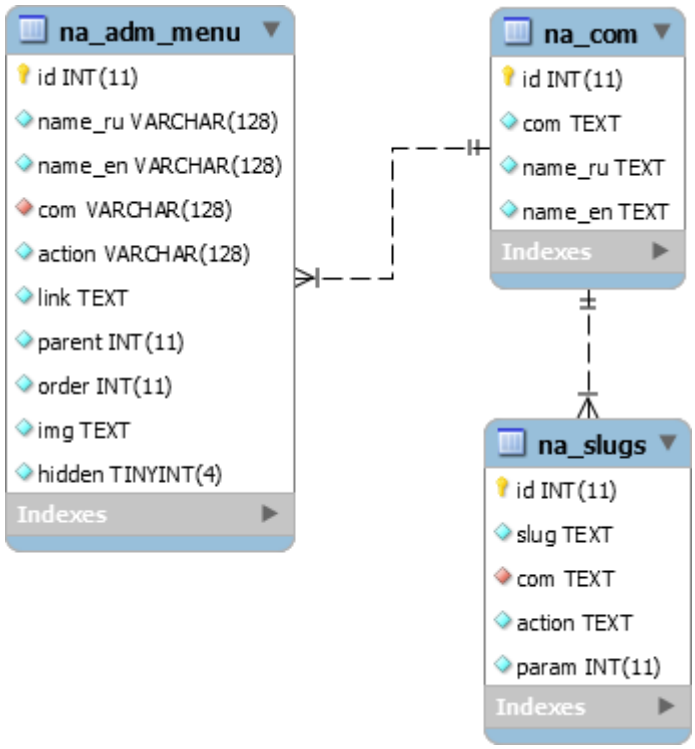


Рисунок 2.6 - Структура таблиць з даними по компонентах та їх взаємозв'язки

а) Загальний список компонентів, що існують:

- ідентифікатор;
- ім'я компонента;
- назва компонента;

б) Набір дій стосовно компоненту для панелі адміністрування:

- ідентифікатор;
- назва дії;
- ім'я компонента;
- дія;
- батьківський елемент;
- порядок сортування;
- стан видимості;

в) Шифрування для відповідної дії в компоненті:

- ідентифікатор;

- алиас (коротке ім’я, нікнейм);

- ім'я компонента;

- дія;

- додатковий параметр.

3) Список користувачів та груп. Для організації управління доступом до панелі адміністрування потрібний базовий набір даних про користувача та можливість розподілу користувачів по групах з урахуванням рівнів доступу кожної групи до будь-яких дій в управлінні контентом вебсайту. В таблиці зі списком користувачів повинна зберігатись наступна інформація (рис.2.7):

- ідентифікатор;

- ПІБ;

- логін;

- email;

- телефон;

- пароль в форматі md5;

- номер групи;

- дата реєстрації;

- стан.

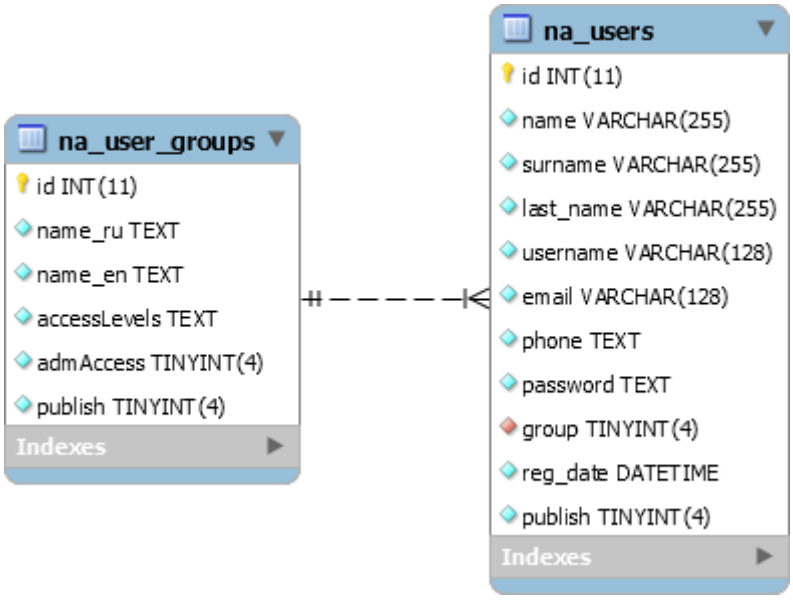


Рисунок 2.7 - Структура таблиць з даними про користувачів

Для кожної групи користувачів потрібно зберігати наступні дані:

- ідентифікатор;

- назва групи;

- набір рівнів доступу групи;

- прапор доступу до панелі адміністрування;

- стан.

4) Набір налаштувань, що зберігається в базі даних може дещо відрізнятись та при потребі редагуватись програмістом, але основні дані для зберігання наступні (рис.2.8):

- email адміністратора;

- SEO дані для головної сторінки сайту (title, keywords, description).

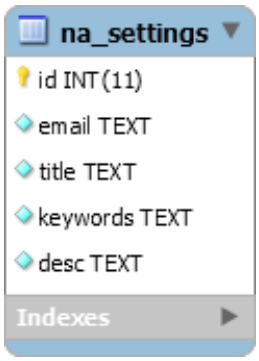


Рисунок 2.8 - Структура таблиці налаштувань

5) Для реалізації невеликих наборів функціональних блоків та спрощення їх виведення в різних місцях шаблону, а також для зниження навантаження при прорахунку сторінки, необхідно реалізувати можливість створення і управління незалежно компільованими програмними модулями, які в динаміці підключаються до основної системи з метою розширення її можливостей. Для забезпечення коректної роботи модулів необхідний такий набір таблиць (рис.2.9):

а) Список існуючих зон для виведення модулів (`na_zones`):

- ідентифікатор;

- шорт-код зони;

б) Список доступних для використання модулів (`na_modlist`):

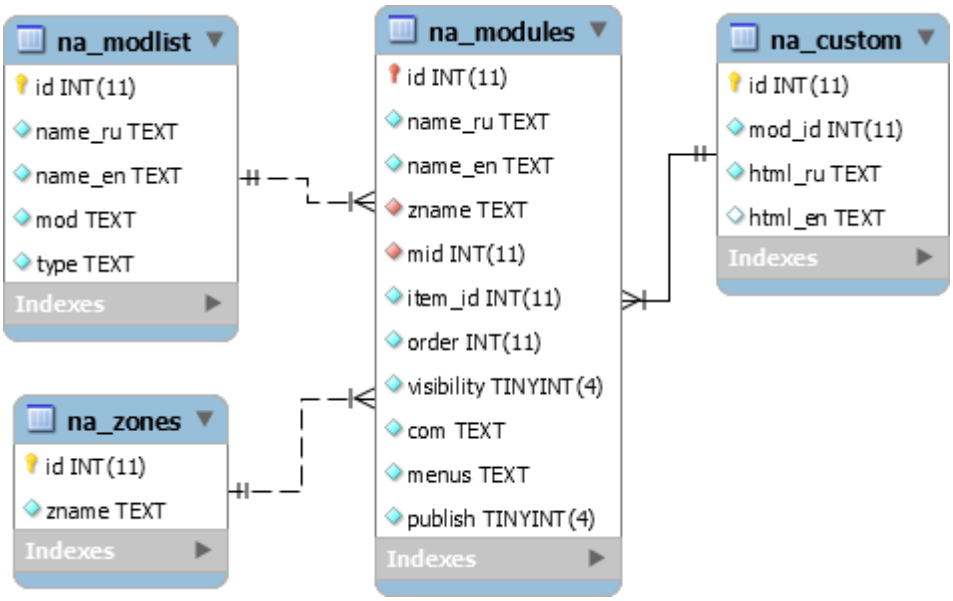


Рисунок 2.9 - Структура таблиць з даними про модулі, що розміщені

- ідентифікатор модуля;
- назва модуля;
- шифр модуля;
- тип модуля;

в) Список розміщених модулів, що настроюються (`na_custom`):

- ідентифікатор запису;
- ідентифікатор модуля;
- набір даних, що зберігається;

г) Список розміщених типізованих модулів (`na_modules`):

- ідентифікатор запису;
- назва модуля;
- зона виведення;
- ідентифікатор модуля;
- порядок виведення;
- прапорець видимості модуля;
- прив'язка до компонентів;
- прив'язка в пунктах меню;
- стан.

2.3 Розробка програмного ядра системи управління вебконтентом

Програмне ядро системи повинне визначати те, як себе буде поводити система в залежності від тих або інших дій користувача. CMS-система запускається, як тільки відвідувач зайшов на вебсайт, що реалізований на базі цієї системи. Точкою входу в систему є файл index.php, що розташований в кореневому каталозі сервера і запускається автоматично при відвідуванні користувачем вебсайту.

Задля забезпечення функціонування системи, всі запити користувачів уразі відвідування будь-якої адреси в межах заданого домену перенаправляються на основну точку входу. Це забезпечують спеціальні налаштування сервера, що зберігаються у конфігураційному файлі .htaccess (рис.2.10).

В точці входу запускаються всі основні функції серверу, та відбувається підключення всіх допоміжних частин ядра системи. Першим підключається конфігураційний файл, в якому визначаються основні константи (даніпідключення до бази даних, префікси таблиць, префікси встановленої мови для клієнтської та серверної частин. Далі визначається тип бази даних та підключається відповідний клас з набором описаних методів для роботи з нею, а також основні класи шаблону MVC (моделі, представлення та контролер). Після того відбувається підключення маршрутизатору та шаблонизатору.

Після підключення необхідних класів, в точці входу відбувається розбір адресного рядка в масив, та виклик ключового методу шаблонизатору для початку побудови вебсторінки.

Основою клієнтської частини (інтерфейсу) є заготовлений шаблон, що містить основний файл шаблону та всі допоміжні файли (стилі, зображення, сценарії та інші). Цей шаблон являє собою каркас головної сторінки вебсайту без інших другорядних сторінок, реалізований мовою гіпертекстової розмітки.

Після підключення усіх ключових файлів ядра здійснюється виклик основного методу шаблонизатору runTemplate (), що запускає початок побудови

вебсторінки. Цей метод робить вибірку з бази даних встановлених шаблонів, визначаючи відповідно до заданих параметрів налаштувань доступний до підключення шаблон. Така схема роботи шаблонизатору дозволяє застосовувати або один, або кілька шаблонів з можливістю їх прикріплення відповідно до заданих критеріїв або до заданої сторінки.

Задля виконання вимог модульності системи розроблений алгоритм, що здійснює підключення та незалежну компіляцію функціональних блоків. Весь функціонал реалізується в модулях та компонентах.

Компоненти - це основні незалежно компільовані блоки, що містять реалізацію основних функціональних можливостей. Для виведення компонентів в контентній зоні шаблону, здійснюється виклик статичного методу маршрутизатору Route::start (), що здійснює запуск самого маршрутизатору.

Всі компоненти та модулі мають однакову структуру - контролер з набором методів обробників подій, модель, а також набір потрібних представлень, тобто шаблон MVC в чистому вигляді.

Після запуску маршрутизатор здійснює пошук і перевірку існування компонента і викликаної дії за запитом користувача (по адресному рядку) та здійснює підключення контролера відповідного компонента та виклик дії.

Кожна дія описує реакцію та поведінку системи у відповідь на той чи інший запит користувача запускаючи при цьому або рендерінг сторінки або обробку отриманих даних з подальшим їх занесенням до бази даних.

Модулі - це невеликі реалізації функціоналу для виведення в конкретних областях шаблону, які не вимагають роботи з великою кількістю представлень та дій, наприклад, виведення блоку новин або коментарів.

Для виведення модуля необхідно в шаблоні у потрібному для виведення місці зробити виклик статичного методу шаблонизатору - Template::genModule ('MZ_ZONENAME'), який в якості вхідного параметра використовує назву зони в верхньому регістрі на латиниці з префіксом MZ. Цей метод здійснює пошук модулю в базі даних за ім’ям зони, виконує підключення контролеру обраного

модулю та виклик методу action_generate () для початку компіляції та рендерінга представлення.

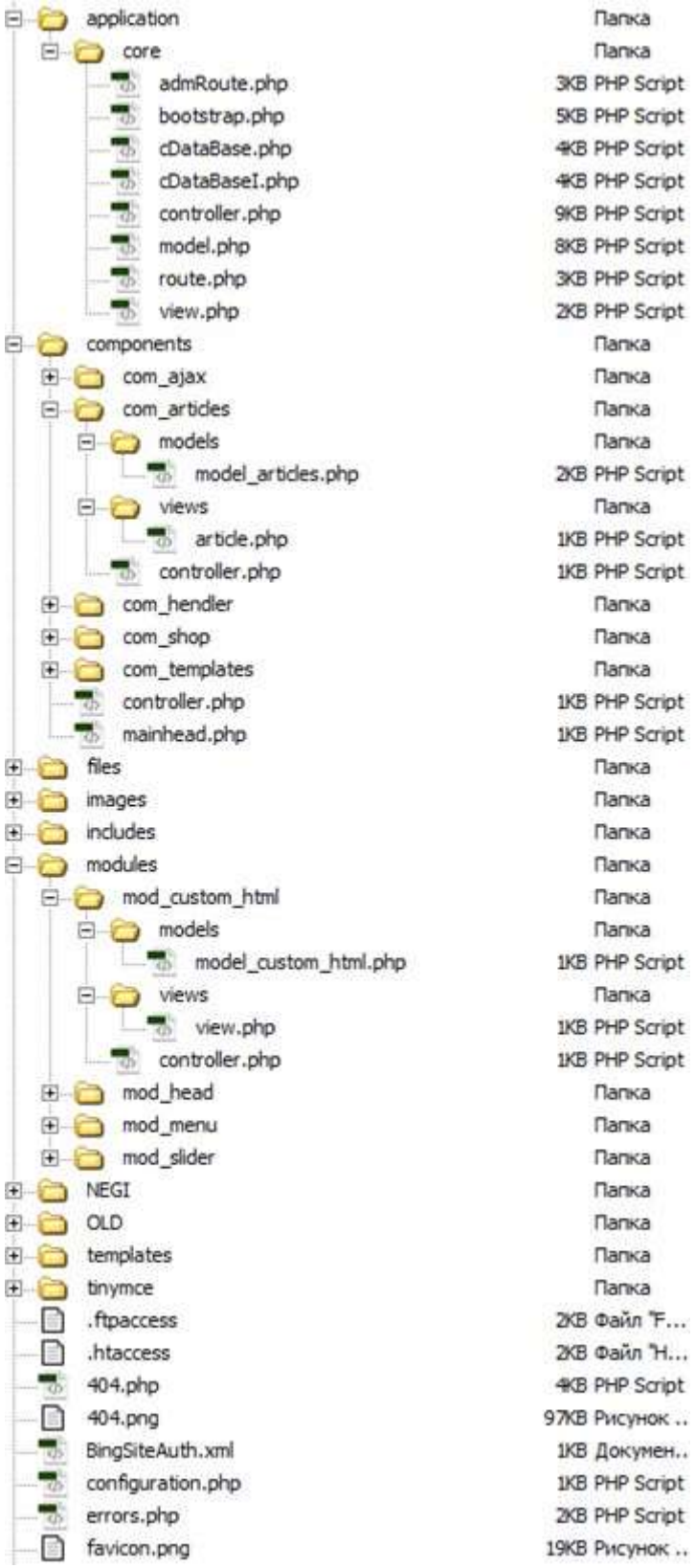


Рисунок 2.10 – Файлова структура системи управління

Імена зон бувають двох типів - або технічні (з префіксом TZ), або модульні (з префіксом MZ). Це реалізовано для відокремлення статичних модулів від функціональних, які в свою чергу можна редагувати і розширювати.

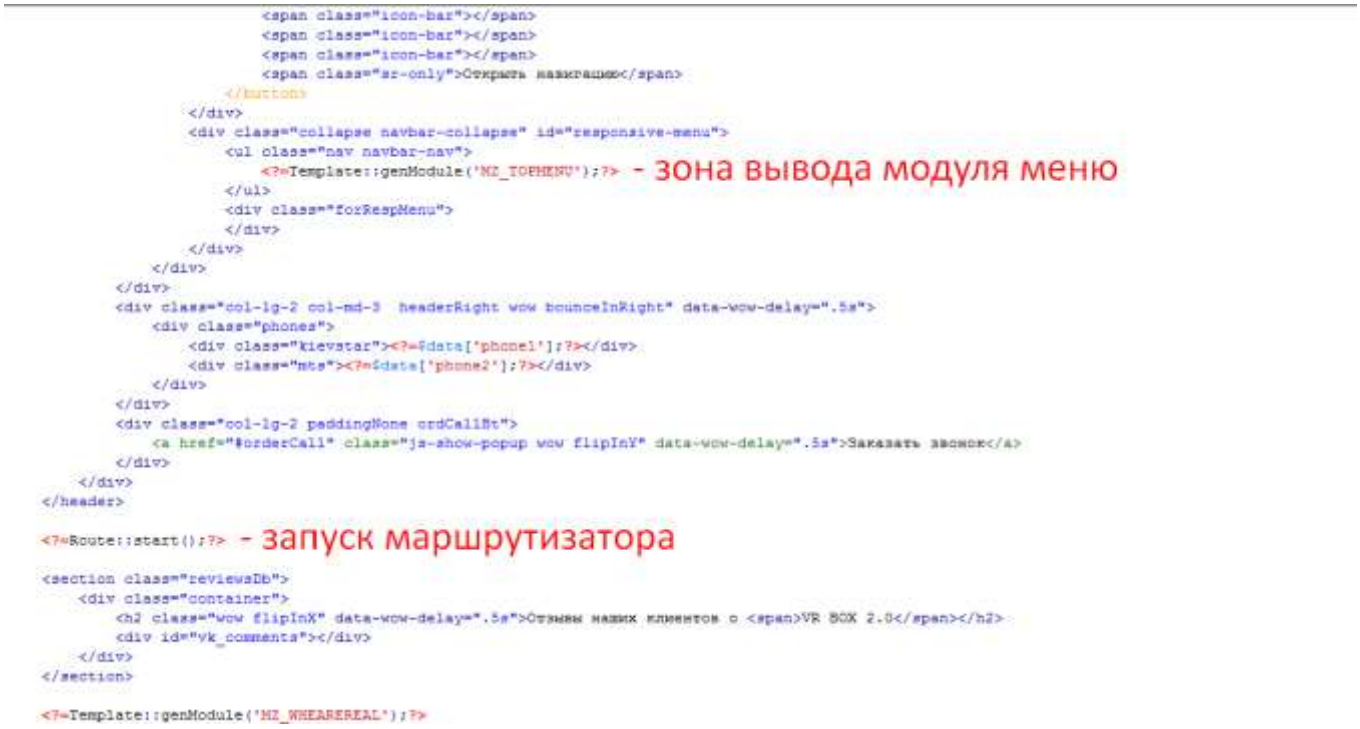


Рисунок 2.11 – Приклад розміщення зон виведення модулів і компонентів

2.4 Створення шаблонів

Для надання розроблюваній системі можливості використовувати різні шаблони потрібно виконати наступні кроки.

Верстку сторінок шаблону розміщуємо в папку з назвою шаблону. В корені цієї папки залишаємо основний файл index.php. Також в корінь папкирозміщуємо файл зображення шаблону у графічному форматі jpg, а також файл інсталяції install.ini (рис.2.12) наступного змісту:

```
name: імя_шаблона
img: імя_зображення
```

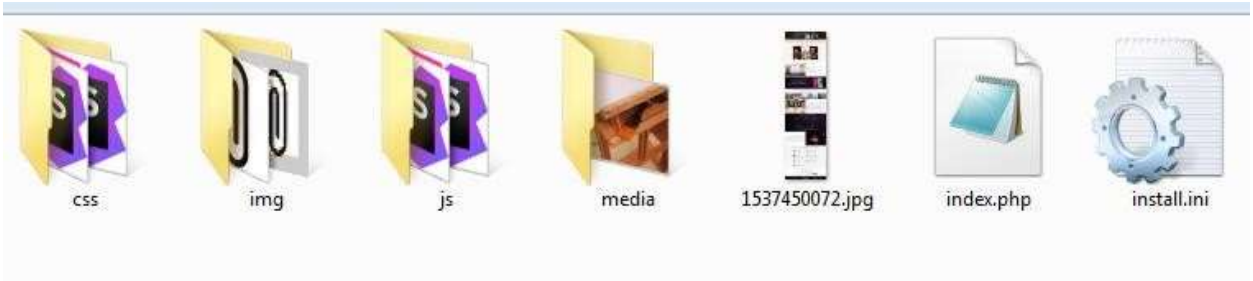


Рисунок 2.12 – Приклад папки для створення шаблону

Наступним кроком необхідно зархівувати усю папку в ZIP архів (рис.2.13).

Рисунок 2.13 – Приклад архівів з шаблонами

Далі створений архів встановлюємо через панель адміністрування системи (рис.2.14).

Рисунок 2.14 – Приклад завантаження шаблонів в системі

Базовий шаблон, що застосовується для головної сторінки вебсайту позначаємо зіркою (рис.2.14).

Кожний шаблон можна налаштувати по-різному, для використання в кожному конкретному випадку - для окремої сторінки, тільки для певного компонента чи компонентів або стану сесії (рис.2.15).

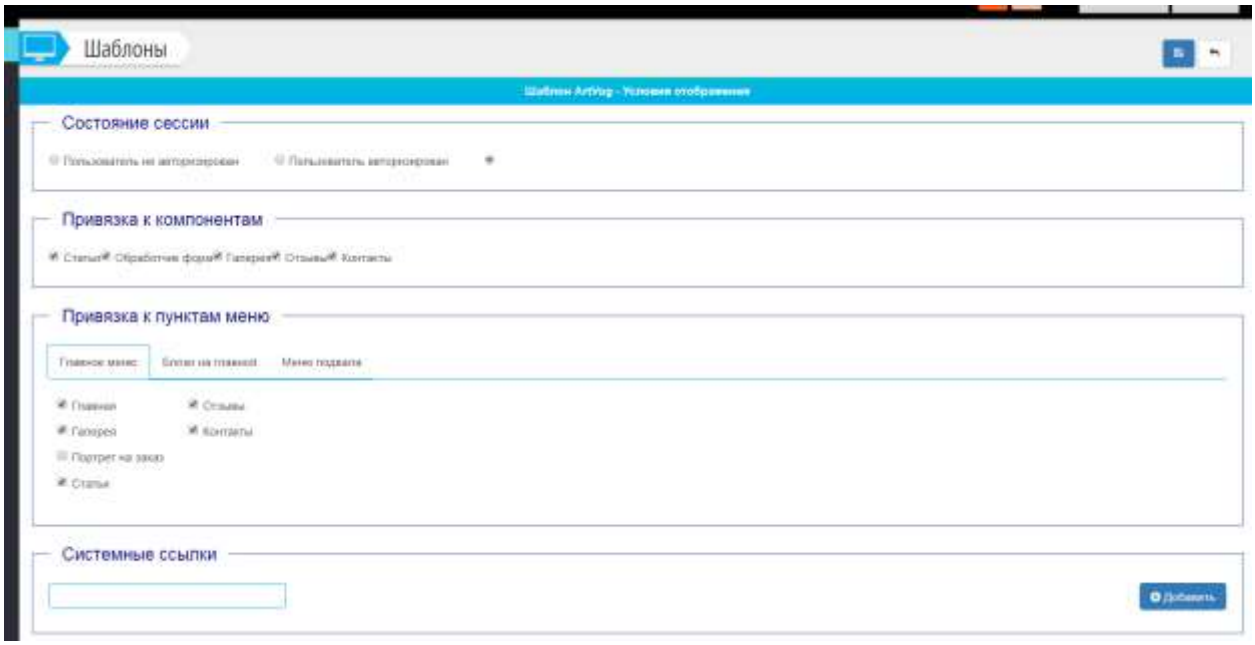


Рисунок 2.15 – Налаштування використання шаблону

Механізм роботи шаблону реалізується наступним чином.

Точкою входу є кореневий файл index.php. Після підключення всіх основних файлів, що потрібні для роботи CMS та сайту в цілому, викликається метод класу Template - runTemplate (), що й реалізує всю логіку (рис.2.16).

```
Route::checkRoute();
if($url[1]!='ajax'  && $url[1]!='handler');
    $template = new Template();
    $template->runTemplate(); //запуск построения страницы
else :
    Route::start();
endif;

ob_flush();
?>
```

Рисунок 2.16 – Представлення View шаблону

Клас Template (Лістинг 2.1) розташований в папці application/core/ bootstrap.php.

Лістинг 2.1 – Клас Template

```
<?php
   class Template extends Controller {
      // Конструктор класу
      function__construct()
      { //Виклик конструктора класу, що унаслідується
         parent::__construct();
      }

      /*
       * Запуск побудови сторінки
       */
      function runTemplate(){
            //Отримуємо кількість доступних шаблонів
            $templates = $this->model->getTemplatesCount();

        // Отримуємо введені користувачем налаштування (для
передачі в шаблон)
            $data = $this->model->getSettings();
            $data['url'] = $this->url;

            if($templates>1) //якщо доступних шаблонів кілька
            {

                //складаємо критерії відбору шаблону
                $condition = "`publish`='1'";
//шаблон повинен бути активний
                $condition .= isset($_SESSION['uid']) ? " AND
`session_status`='1'" : " AND `session_status`='0'";
//перевіряємо стан сесії

         // Отримуємо ім'я компонента по URL в адресному рядку
                $slug = !empty($this->slug{com}) ? $this-
>slug{com} :  $this->url{1};

                //вказуємо критерії пошуку по імені компонента
і поточною адресою сторінки
                $condition .= !empty($this->url{1}) ? " AND
`com` LIKE '%".$slug."%' OR `menu` LIKE
'%".$_SERVER['REQUEST_URI']."%' OR `links` LIKE
'%".$_SERVER['REQUEST_URI']."%'" : " AND `status`='1'";

                //робимо вибірку шаблону за вказаними
налаштуваннями
                $templates = $this->model-
>getTemplatesList($condition);
```

```
                //підключаємо шаблон
                $this->view-
>genTemplate($templates[0]['name'],$data);
            }else{
                // якщо доступний тільки один шаблон
                $templates = $this->model-
>getTemplatesList("`publish`='1' AND `status`='1'");
                $this->view-
>genTemplate($templates[0]['name'],$data);
            }
        }
    }
```

View Controller - загальний контролер представлень (Лістинг 2.2).

Лістинг 2.2 – Загальний контролер уявлень

```
// Генерація основного шаблона для front-end
function genTemplate($template_view,$data=null)
{
   include 'templates/'.$template_view.'/index.php';
}
```

3 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ УПРАВЛІННЯ

3.1 Тестування функціоналу системи управління

Панель керування надає набір інструментів для управління вмістом клієнтської частини (front-end) вебсайту. Увесь її функціонал базується на реалізації набору компонентів, що відповідають за управління тими або іншими даними. Панель містить базовий функціонал для управління, але компоненти також розробляються виходячи з конкретних вимог до проекту. Компоненти працюють за тим же принципом, але маршрутизація відбувається вже шляхом $_GET запитів, що полегшує маршрутизатор та спрощує його структуру.

Для реалізації панелі управління першочерговим завданням є обмеження доступу та захист її від несанкціонованого доступу. З цією метою для переходу в панель адміністрування користувачу необхідно авторизуватись в системі.

При авторизації також надається можливість вибору мови інтерфейсу (рис.3.1). Система поки що підтримує дві мови - російську та англійську.

Рисунок 3.1 – Форма авторизації для входу до панелі адміністрування

Виходячи з вимог технічного завдання, в розробленій системі була реалізована підтримка багато-користувацького режиму з можливістю управління доступом користувачів до одних або інших дій щодо керування налаштуванням системи та контентом вебсайту.

Відповідно, однією з базових функцій системи є можливість за допомогою панелі адміністрування додавання нових користувачів та нових груп користувачів з розподілом користувачів за групами.

Додати нового користувача можна в пункті меню Користувачі>Список користувачів>Створити користувача (рис.3.2), заповнюючи у формі всі необхідні поля, а також обравши групу до якої буде цей користувач належати. З метою забезпечення безпеки і збереження даних цю можливість потрібно надати виключно групі адміністраторів.

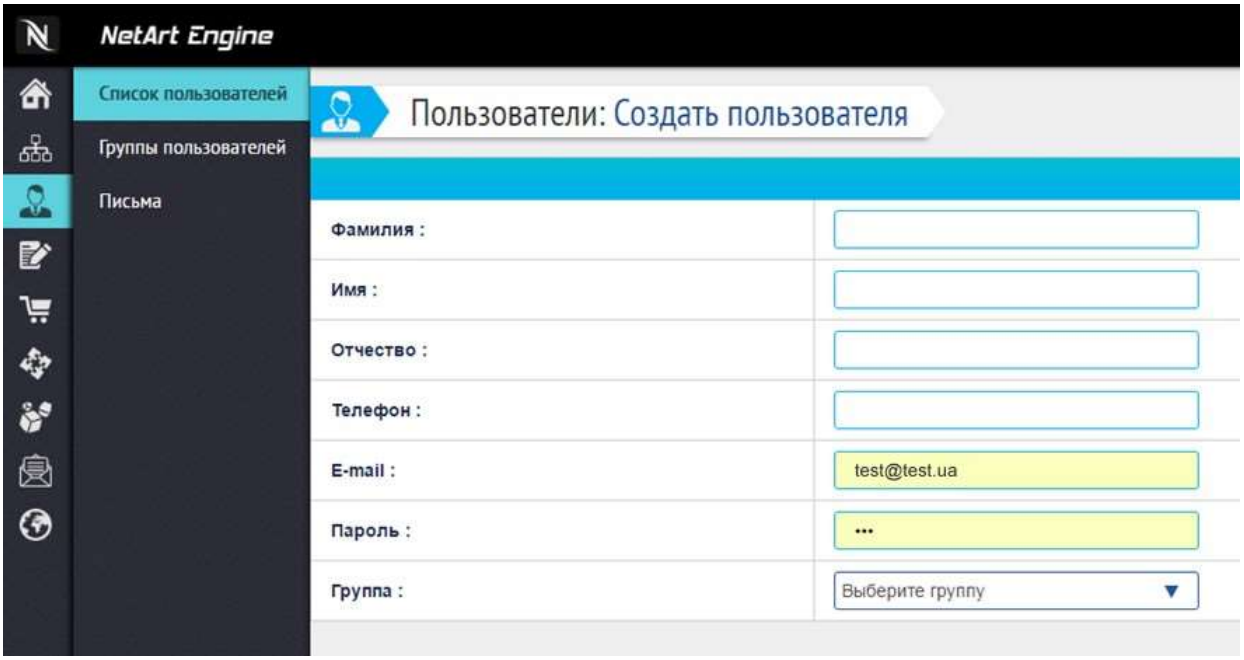


Рисунок 3.2 - Додавання нового користувача

Для налаштування рівня доступу конкретної групи та, відповідно, обмеження доступу користувачів цієї групи в їх можливих діях, необхідно обрати пункт меню Користувачі> Групи користувачів>Рівні доступу групи (рис.3.3).

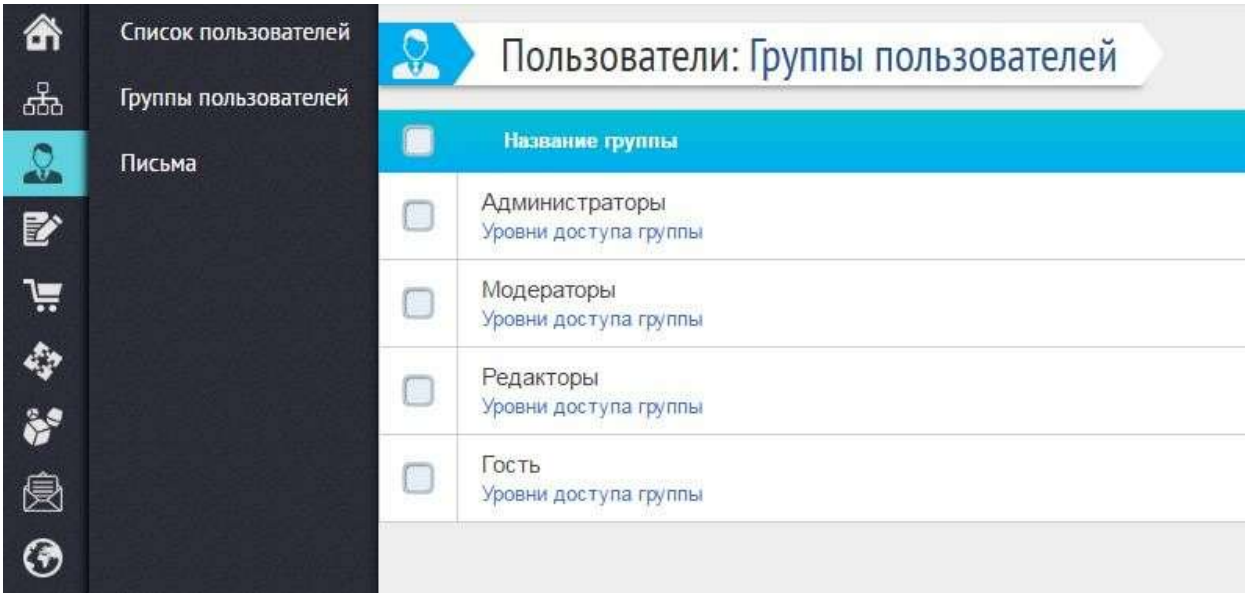


Рисунок 3.3 - Управління групами користувачів

В рівнях доступу групи в окремому вигляді за кожним компонентом відображений список доступних дій (рис.3.4). Для обмеження доступу достатньо зняти галочку з відповідної дії та зберегти зміни. Також реалізована можливість дозволу або заборони доступу до панелі управління в цілому для конкретноїгрупи. Цей інструмент є особливо корисним, якщо вебсайт достатньо великого обсягу з яким працює багато користувачів різного профілю, наприклад, редактори, модератори, SEO-оптимізатори та інші.

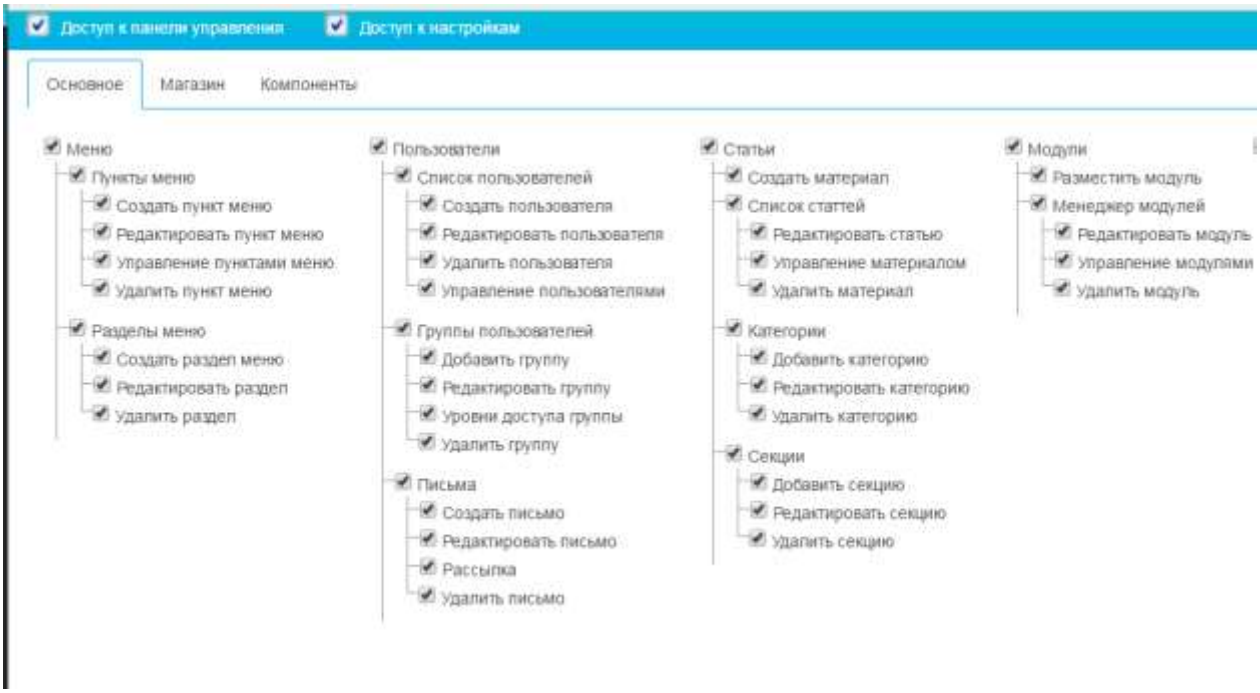


Рисунок 3.4 - Управління рівнями доступу груп користувачів

З метою більш зручного управління модулями розроблений відповідний окремий компонент, що дає змогу розмежувати області видимості модулів, а також дозволяє досить просто їх розміщувати, редагувати, управляти відображенням та порядком виведення. При переході до менеджеру модулів користувачу стає доступним список вже розміщених модулів (рис.3.5) з описом їх типу, зони виведення то областю видимості. Більш детальну інформацію щодо модуля можна побачити в процесі його редагування (рис.3.6).

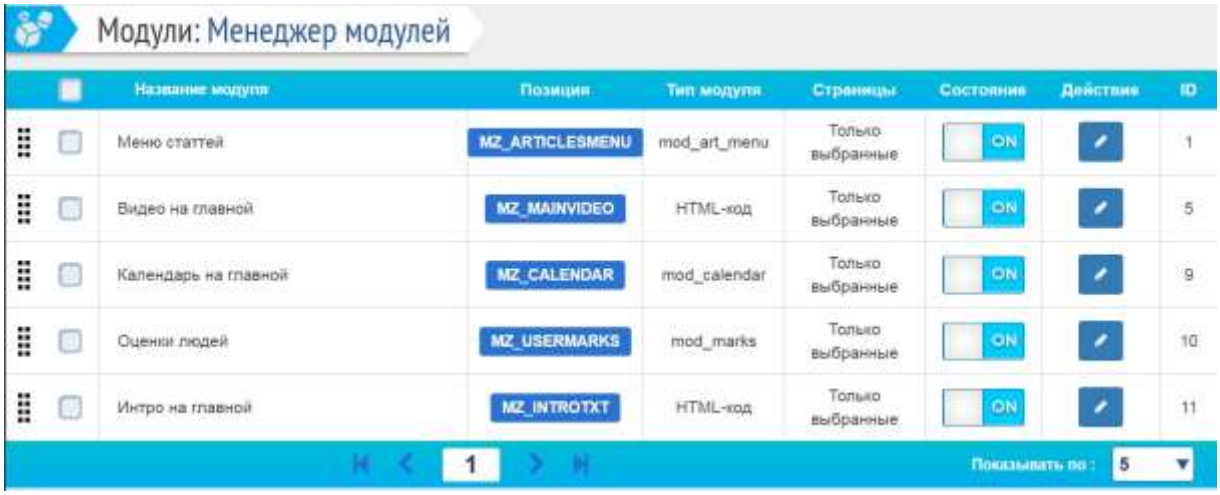


Рисунок 3.5 - Менеджер модулів

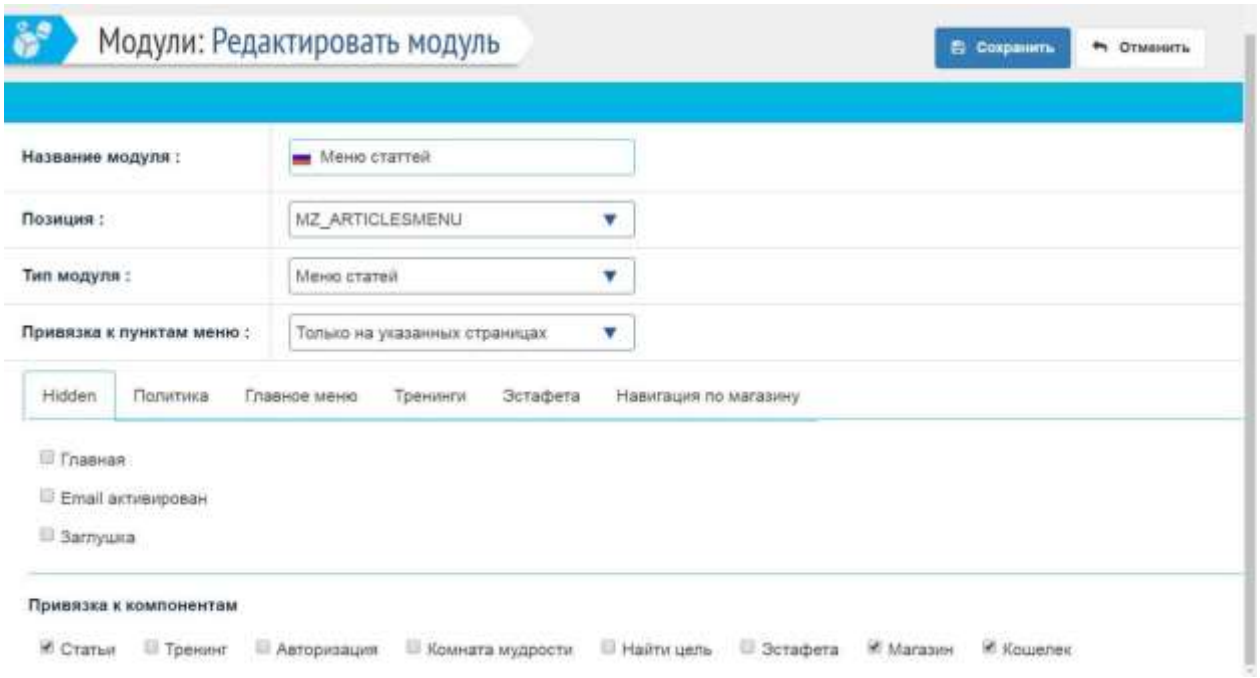


Рисунок 3.6 - Управління модулем

3.2 Тестування підключення шаблонів

Також однією з головних вимог до розроблюваної системи є підтримка різних шаблонів. Для цього в складі системи має бути компонент, що надає

можливість встановлювати шаблони та керувати ними за допомогою панелі управління.

Для встановлення шаблону необхідний набір основних файлів шаблону, файл точки входу index.php (зазвичай головна сторінка), інсталяційний файл install.ini з набором налаштувань зібрані та стиснуті в архів розширення .zip.

Щоб встановити шаблон в систему, в панелі необхідно його обрати та завантажити, після чого на сервері відбувається розпакування даного архіву та інсталяція відповідно до набору інструкцій.

Усі встановлені шаблони відображаються в панелі адміністрування Компоненти>Шаблони і мають ряд налаштувань (рис.3.7). Завжди потрібно обрати основний шаблон, що буде використовуватися за замовчуванням для головної сторінки вебсайту, а при необхідності і для інших сторінок. При бажанні шаблон можна відключити без його безпосереднього видалення шляхом зміни його стану (зняти з публікації).

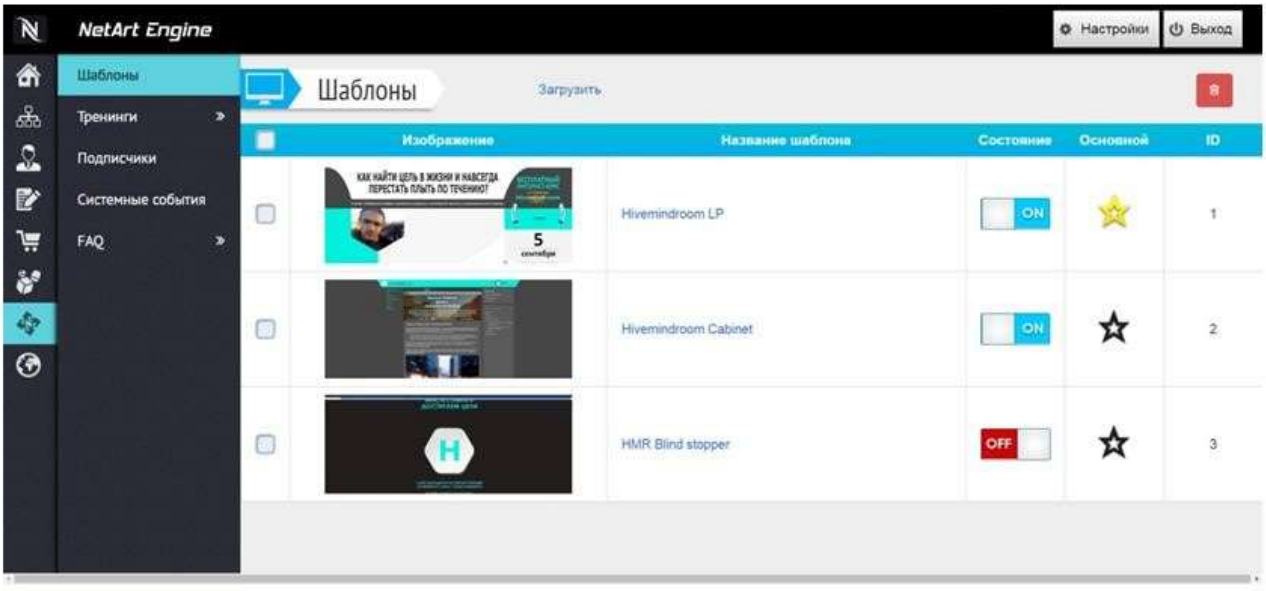


Рисунок 3.7 - Управління встановленими шаблонами

У кожного шаблону є набір налаштувань, що дозволяє прив'язати його застосування до компоненту, до необхідного пункту меню або навіть доконкретної адреси на вебсайті (рис.3.8). Це дозволяє більш гнучка управляти великою кількістю шаблонів та їх застосуванням.

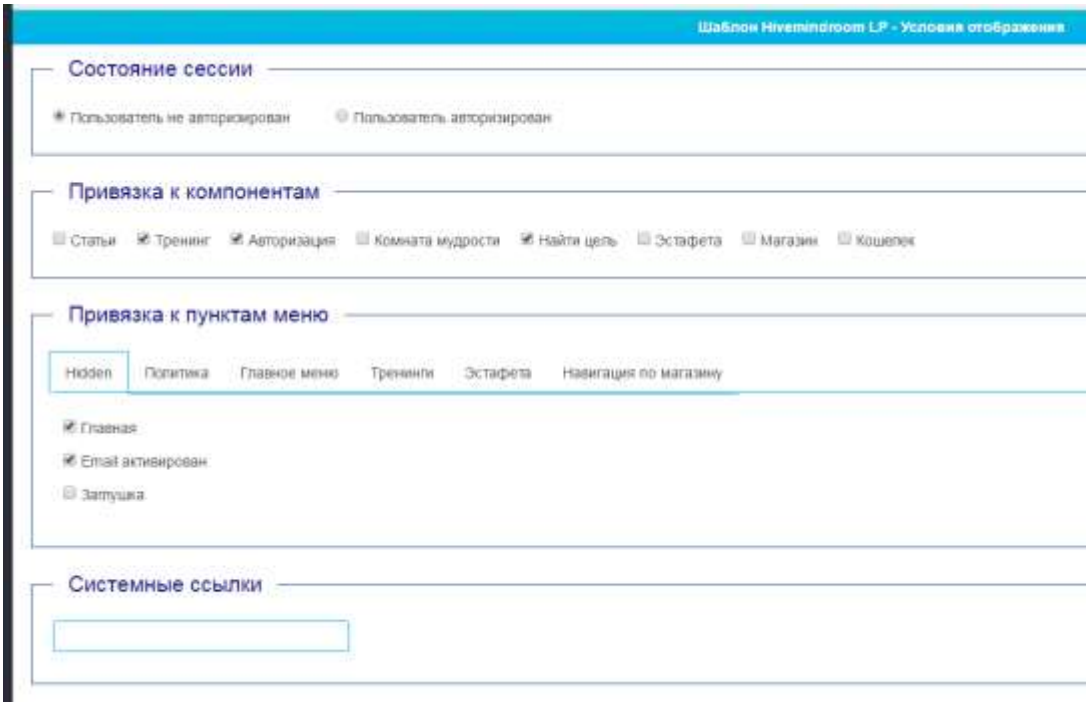


Рисунок 3.8 - Налаштування застосування шаблону

3.3 Тестування продуктивності розробленої системи

Для порівняння продуктивності розробленої системи, проведемо тестування вебсайту, створеного за допомогою розробленої системи управління вебконтентом та одного з популярних сайтів, що створений на CMS WordPress.

Тестування проводимо за допомогою застосунку з набору для розробника від компанії Google - Lighthouse [12].

Lighthouse - це автоматизований інструмент з відкритим вихідним кодом для покращення якості вебсторінок. Його можна запустити на будь-який вебсторінці, загальнодоступній або такої, що вимагає аутентифікації. Він проводить аудит продуктивності, доступності, прогресивних веб-застосунків, SEO та багато чого іншого.

Ми даємо Lighthouse URL-адресу для аудиту, він запускає серію аудитів сторінки, а потім генерує звіт про те, наскільки добре сторінка працює. Невдалі аудити можна використовувати як індикатори того, як покращити вебсторінку. У

кожного аудиту є довідкова документація, в якій пояснюється, чому аудит важливий, а також як його виправити.

Lighthouse можна запустити наступними способами:

- у Chrome DevTools. Легко перевіряти вебсторінки, що вимагають аутентифікації, і читайте звіти в зручному для користувача форматі;

- з командного рядка. Можна автоматизувати запуск Lighthouse за допомогою сценаріїв оболонки;

- як модуль вузла. Можна інтегрувати Lighthouse в свої системи безперервної інтеграції;

- з вебінтерфейсу. Запускаємо Lighthouse і посилаємося на звіти, не встановлюючи нічого.

Для простоти та зручності був застосований саме останній варіант:

- переходимо на сторінку PageSpeed Insights;

- вводимо URL-адресу вебсторінки;

- клацаємо "Аналізувати".

Причому налається можливість аналізу як для мобільних пристроїв, так і для комп'ютерів (рис.3.9).

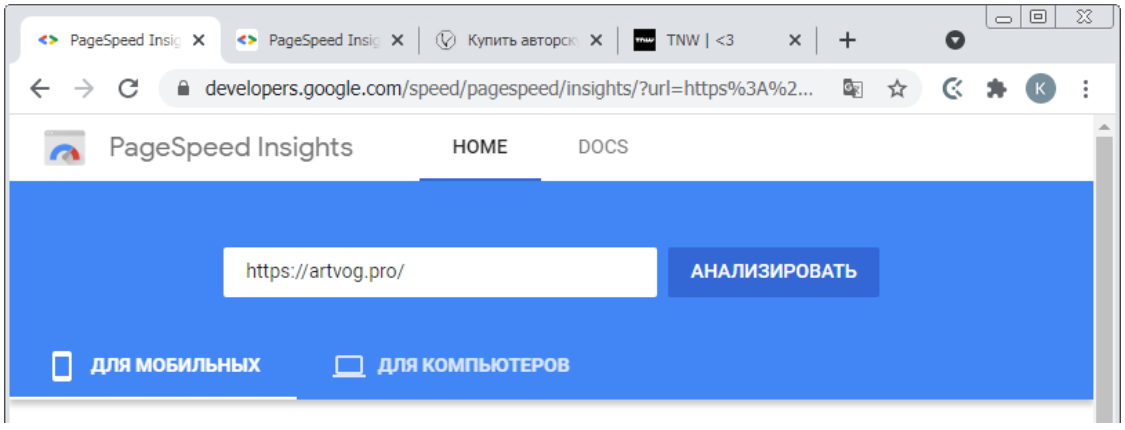


Рисунок 3.9 – Запуск вебсторінки для аналізу в Lighthouse

Першим тестуємо вебсайт, створений за допомогою розробленої системи управління. Отримуємо наступні результати тестування для мобільних пристроїв (рис.3.10) та комп'ютерів (рис.3.11).

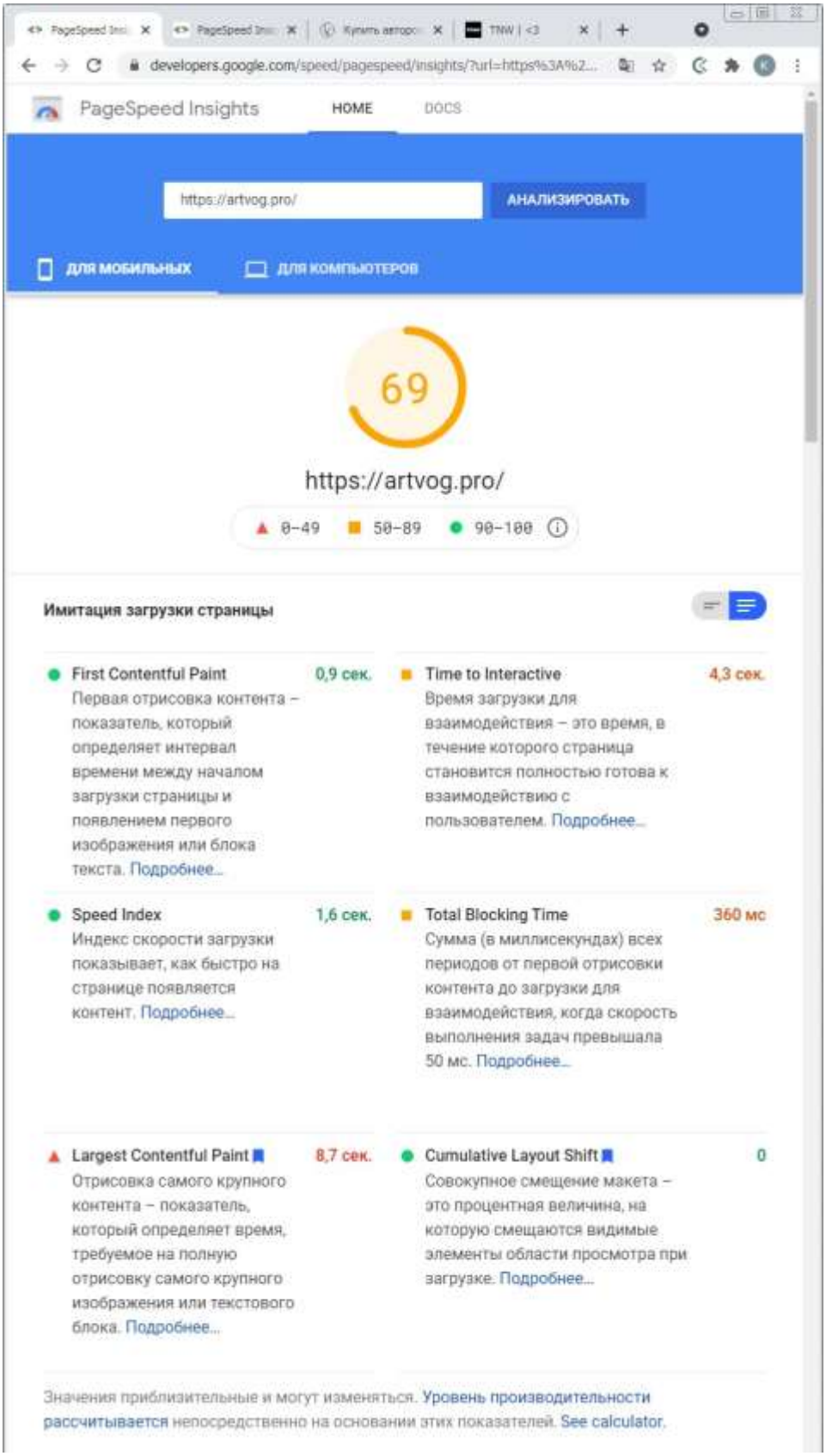


Рисунок 3.10 – Результати тестування для мобільних пристроїв

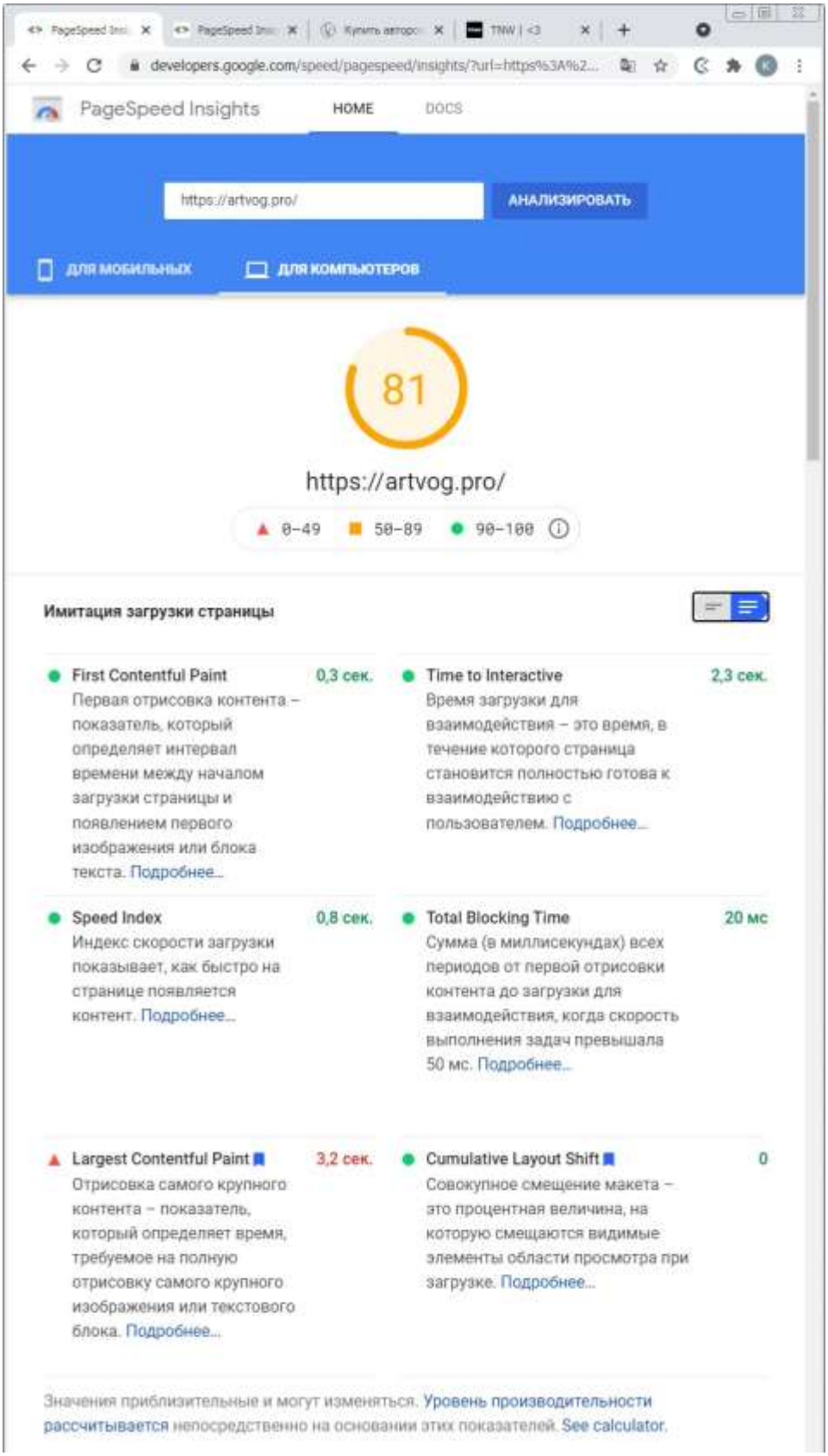


Рисунок 3.11 – Результати тестування для комп’ютерів

Для порівняння був обраний самий популярний вебсайт, створений на базі CMS WordPress – сайт самої системи wordpress.com. Отримуємо такі результати його тестування для мобільних пристроїв (рис.3.12) та комп’ютерів (рис.3.13).

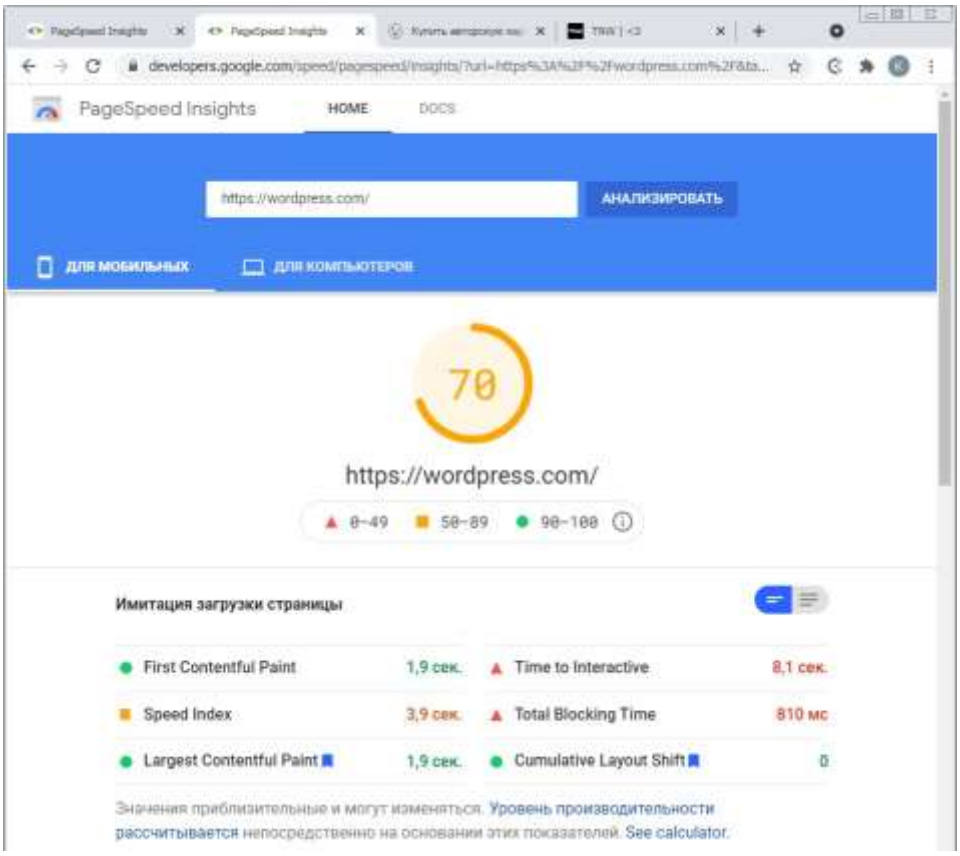


Рисунок 3.12 – Результати тестування для мобільних пристроїв

За результатами порівняння отриманих результатів можна зробити висновок, що для мобільних пристроїв системи працюють приблизно однаково. Деяка перевага існує у системи WordPress при роботі на комп’ютері, але й показники розроблювальної системи є теж досить непоганими.

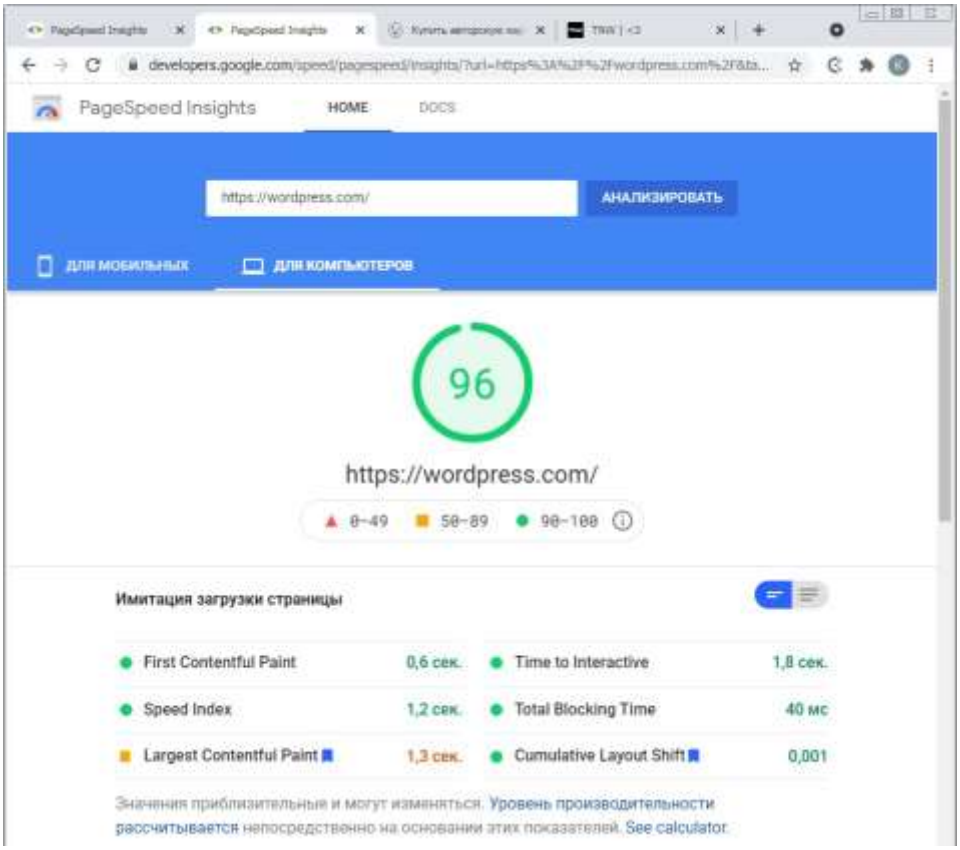


Рисунок 3.13 – Результати тестування для комп’ютерів

ВИСНОВКИ

В процесі дипломної роботи була розроблена повноцінна система управління вебконтентом з базовим набором функцій, достатньо зручним та простим інтерфейсом.

Була проаналізований функціонал існуючих систем управління контентом, технології та програмні засоби розробки вебсайтів та вебзастосунків. На підставі цього аналізу було прийняте рішення створити власну систему.

При розробці системи управління було вирішено наступні завдання:

- спроектовано і реалізовано базу даних системи;
- реалізована функція реєстрації та авторизації користувачів в системі;
- розроблено особистий кабінет користувача з можливістю перегляду зроблених замовлень;
- створена проста навігаційна панель на основі категорій товарів;
- створено компонент каталогу з виведенням категорій і товарів;
- спроектовано та реалізовано функцію підключення додаткових шаблонів.

Після розробки системи було проведено тестування її функціоналу та продуктивності, що підтвердило її конкурентоспроможність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Системы управления контентом [Електронний ресурс]. – Режим доступу: https://intuit.ru/studies/professional_skill_improvements/1450/courses/239/lecture/6178?page=2 - Заголовок з екрану.

2. Исследование популярности CMS за 2021 год [Електронний ресурс]. – Режим доступу: https://itrack.ru/research/cmsrate/#!cms-free-tab - Заголовок з екрану.

3. Usage statistics of server-side programming languages for websites [Електронний ресурс]. – Режим доступу: https://w3techs.com/technologies/overview/programming_language - Заголовок з екрану.

4. Шаблон Bootstrap [Електронний ресурс]. – Режим доступу:/ http://bootstrap-ru.com/scaffolding.php - Заголовок з екрану.

5. Клименко, Р. Веб-мастеринг на 100% / Роман Клименко. – СПб.: Питер, 2013. – 512 с.

6. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Робин Никсон. Пер. с англ. – СПб.: Питер, 2016. – 767 с.

7. Хопкинс, К. PHP. Быстрый старт / Каллум Хопкинс. Пер. с англ. – М.: Эксмо, 2014. – 159 с.

8. Флэнаган, Д. JavaScript. Подробное руководство, 6-е издание / Дэвид Флэнаган. Пер. с англ. – М.: Символ-Плюс, 2017. – 1080 с.

9. Гольцман, В. MySQL 5.0. Библиотека программиста / Виктор Гольцман – СПб.: Питер, 2010. – 230 с.

10. Веллинг, Л. Разработка веб-приложений с помощью PHP и MySQL, 5-е издание / Люк Веллинг, Лора Томсон. – М: Вильямс, 2017. - 768 стр.

11. Руководство по проектированию реляционных баз данных [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/194714/ - Заголовок з екрану.

12. Lighthouse. Tools for Web Developers [Електронний ресурс]. – Режим доступу: https://developers.google.com/web/tools/lighthouse - Заголовок з екрану.

13. Методичні вказівки до виконання дипломної роботи бакалавра для студентів спеціальності 123 «Комп’ютерна інженерія» всіх форм навчання / укл.: Р. К. Кудерметов, О. В. Польська, Н. В. Луценко, Н. В. Щербак, О. В. Зелік. Запоріжжя : НУ «Запорізька політехніка», 2020. 54 с.

Додаток А
ПРОГРАМНИЙ КОД ЗАСТОСУНКУ

Лістинг А.1 – Точка входу (index.php)

```
<?php
ini_set('display_errors', '0');
error_reporting( E_ERROR );
session_start();
ob_start();

if($_SERVER['REQUEST_URI'] == "/index.php" ||
$_SERVER['REQUEST_URI'] == "/index.html" ||
$_SERVER['REQUEST_URI'] == "/index"):
header("Location: /",TRUE,301);
exit();
endif;

require_once 'configuration.php';
$cDataBase = DBTYPE == 'mysqli' ? 'cDataBaseI.php' :
'cDataBase.php';
require_once 'application/core/'.$cDataBase;
require_once 'application/core/model.php';
$url = explode("/",$_SERVER['REQUEST_URI']);

$homepage =
file_get_contents('http://'.$_SERVER['HTTP_HOST'].'/includes/l
ang'.ADMLANG.'.xml');
$dom = new domDocument;
$dom->loadXML($homepage);
$GLOBALS = simplexml_import_dom($dom);

require_once 'application/core/view.php';
require_once 'application/core/controller.php';
require_once 'application/core/route.php';     //Роутер
компонентов
require_once 'application/core/bootstrap.php'; //Конструктор
темплаты

Route::checkRoute();
if($url[1]!='ajax'):
$template = new Template();
$template->runTemplate(); //запуск построения страницы
else :
Route::start();
endif;

ob_flush();
?>
```

Лістинг А.2 – Файл конфигурації (configuration.php)

```
<?php
define("DBHOST","localhost");
define("DBUSER","narek");
define("DBPASSW","narek");
define("DBNAME","blshop");
define("DBPREFIX","na_");
define("DBTYPE","mysql");

define('HTTP_SERVER', 'http://blshop.com.ua/');
define('HTTPS_SERVER', 'https://blshop.com.ua/');

define("HOST",substr($_SERVER['HTTP_HOST'], -
1)=='/'?$_SERVER['HTTP_HOST']:$_SERVER['HTTP_HOST'].'/');
define("BASEURL",'http://'.HOST);
define("BASEDIR",substr($_SERVER['DOCUMENT_ROOT'], -
1)=='/'?$_SERVER['DOCUMENT_ROOT']:$_SERVER['DOCUMENT_ROOT'].'/
');
define('ROOT',dirname(__FILE__));

define("ADMLANG",isset($_COOKIE['admLang']) ?
'_'.$_COOKIE['admLang'] : '_ru');
define("FRONTLANG",isset($_COOKIE['lang']) ?
'_'.$_COOKIE['lang'] : '_ru');
if(isset($_SESSION['user_id']) &&
!empty($_SESSION['user_id']))
define("USER",$_SESSION['user_id']);
?>
```

Лістинг А.3 – Класс для работы с БД

```
<?php
class CDataBase
{
var $link;
var $db;
function__construct()
{
$this->link=mysql_connect(DBHOST,DBUSER,DBPASSW)or die
(mysql_error());
if (!$this->link)return 0;
$this->db=mysql_select_db(DBNAME,$this->link)or die
(mysql_error());
if (!$this->db)return 0;
mysql_query ("SET NAMES utf8",$this->link);
mysql_query ("set character_set_client='utf8'",$this->link);
mysql_query ("set character_set_results='utf8'",$this->link);
mysql_query ("set
collation_connection='utf8_general_ci'",$this->link);

}
```

```
function SendQuery($query)
{
$res = mysql_query ($query,$this->link)or
die(mysql_error()."<br>".$query);
if (!$res) return 0; else return $res;
}

function clearData($var){

$var = strip_tags($var);
$var = trim($var);
$var = htmlspecialchars($var);
$var = mysql_real_escape_string($var);

return $var;
}

function getRow($query){
$res = mysql_query ($query,$this->link)or
die(mysql_error()."<br>".$query);
if (!$res) return 0; else return mysql_fetch_assoc($res);
}

function insert($table,$cols = null,$values = null){
mysql_query ("INSERT INTO `".DBPREFIX.$table."`
(".implode(",",$cols).")
VALUES(".implode(",",$values).");",$this->link)or
die(mysql_error()."<br>".$query);
}

function getAssoc($rows,$table,$ordby = null,$ordtype =
null,$where=null,$limit = null){
if(!empty($ordby))
$order = ' ORDER BY `'.$ordby.'` '.$ordtype;
if(!empty($where))
$where = " WHERE ".$where."";

if(!empty($limit))
$limit = " LIMIT ".$limit."";

$query = mysql_query ("SELECT ".$rows." FROM
".DBPREFIX.$table.$where.$order.$limit."",$this->link)or
die(mysql_error());
$data = array();
while($res = mysql_fetch_assoc($query))
{
$data[] = $res;
}

if (!$data) return 0; else return $data;
}
```

```
function getRowAssoc($res){
$data = mysql_fetch_assoc($res);
return $data;
}

function getResult($res){
$data = mysql_result($res,0);
return $data;
}
function getLast(){
$id = mysql_insert_id();
return $id;
}

function delete($table, $id, $col = null)
{
if($table!='languages')
{
$col = !$col ? 'id' : $col;
foreach($id as $mas_id)
{
$this->SendQuery("DELETE FROM `".DBPREFIX.$table."` WHERE
`".$col."`='".$mas_id."';");
}
}else{
foreach($id as $mas_id)
{
$this->SendQuery("DELETE FROM `".DBPREFIX.$table."` WHERE
`".$col."`='".$mas_id."' AND `status`!='1';");
}
}
}
function publish($table, $id,$colName=null)
{
$colName = isset($colName) ? $colName : 'id';
$this->SendQuery("UPDATE `".DBPREFIX.$table."` SET
`publish`='1' WHERE `".$colName."`='".$id."';");
}

function unpublish($table, $id,$colName=null)
{
$colName = isset($colName) ? $colName : 'id';
$this->SendQuery("UPDATE `".DBPREFIX.$table."` SET
`publish`='0' WHERE `".$colName."`='".$id."';");
}

function activateStatus($table,$id){
$this->SendQuery("UPDATE `".DBPREFIX.$table."` SET
`status`='0';");
$this->SendQuery("UPDATE `".DBPREFIX.$table."` SET
`status`='1',`publish`='1' WHERE `id`='".$id."';");
}
```

```
function updateItem($table,$col,$id,$val,$colName=null){
$colName = !empty($colName) ? $colName : 'id';
$this->SendQuery("UPDATE `".DBPREFIX.$table."` SET
`".$col."`='".$val."' WHERE `".$colName."`='".$id."';");
}

function escapeString($val){
$data = mysql_real_escape_string($val);
return $data;
}
}
?>
```

Лістинг А.4 – Ядро системи

```
<                    Model                    >
<?php
class Model
{
public $db_Object; // свойство доступа к дейтсвиям над БД

function___construct()
{
$this->db_Object = new CDataBase();
}

public function get_data()
{
}

/*
* Модель данных для бокового меню
* панели администратора
*/
public function get_leftBar($parent){
$items = $this->db_Object->SendQuery("SELECT * FROM
`".DBPREFIX."adm_menu` WHERE `parent`='$parent' AND
`hidden`='0' ORDER BY `order` ASC;");

$item_list = array();
while($res = $this->db_Object->getRowAssoc($items))
{
$item_list[] = $res;
}
return $item_list;
}

public function getLBParent($action){
$data = $this->db_Object->getRowAssoc($this->db_Object-
>SendQuery("SELECT * FROM `".DBPREFIX."adm_menu` WHERE `id` IN
(
```

```
SELECT `parent` FROM `".DBPREFIX."adm_menu` WHERE
`com`='".$_GET['com']."' AND `action`='$action'
);"));

return $data;
}

/*
* Модель данных для имен
* компонентов и действий (back-end)
*/
public function get_Title($com,$action){

$title[0] = $this->db_Object->getRow("SELECT `name".ADMLANG."`
FROM `".DBPREFIX."adm_menu` WHERE `com`='$com' LIMIT 1;");
$title[1] = $this->db_Object->getRow("SELECT `name".ADMLANG."`
FROM `".DBPREFIX."adm_menu` WHERE `com`='$com' AND
`action`='$action' ORDER BY `id` DESC LIMIT 1;");
return $title;
}

/*
* Модель данных получения
* префикса основного языка для front-end
*/
function getMainLang(){
$data = $this->db_Object->getRow("SELECT `prfix` FROM
`".DBPREFIX."languages` WHERE `status`='1' AND `publish`='1'
LIMIT 1;");
$prfix =$data['prfix'];
return $prfix;
}

/*
* Модель данных получения списка
* префиксов языков для front-end
*/
function getPrefixList($condition=null){
$where = !empty($condition) ? $condition : '';
$data = $this->db_Object-
>getAssoc("prfix","languages",'','',$where);
return $data;
}

/*
* Модель данных получения списка
* языков для front-end
*/
function getLanguagesList($condition=null){
$where = !empty($condition) ? $condition : '';
$data = $this->db_Object-
>getAssoc("*","languages","status","DESC",$where);
```

```
return $data;
}

/*
* Модель данных получения списка
* языков для панели администратора (back-end)
*/
function getAdmLangList($condition=null,$value=null){
$where = !empty($condition) ? "`".$condition."`='".$value."'"
: '';
$data = $this->db_Object-
>getAssoc("*","adm_languages",'','',$where);
return $data;
}

/*
* Модель данных уровней доступа группы (back-end)
*/
function getGroupAccessLevels($group){
$data = $this->db_Object-
>getAssoc('`accessLevels`','user_groups','','',"`id`='$group'"
);
if($data)
{
$data = json_decode($data[0]['accessLevels'],true);
}

return $data;
}

/*
* Модель получения идентификатора
* текущего действия (back-end)
*/
function getActionId($com,$action,$condition=null){

if(!empty($condition))
$where = " AND `parent` IN (SELECT `id` FROM
`".DBPREFIX."adm_menu` WHERE `action`='".$condition."')";

$data = $this->db_Object-
>getAssoc('`id`','adm_menu','','',"`com`='$com' AND `action`
LIKE '%".$action."%'".$where);
if($data)
{
$ids = array();
foreach($data as $id)
{
$ids[] = $id['id'];
}
}
```

```
return $ids;
}

/*
* Модель получения конкретного
* столбца из БД
*/
function getColItem($table,$column,$id,$colName=null){
$colName = !empty($colName) ? $colName : 'id';
$data = $this->db_Object-
>getAssoc('`'.$column.'`',$table,'','',"`".$colName."`='$id'")
;
return $data[0][$column];
}

function getTemplatesCount(){
$data = $this->db_Object->getResult($this->db_Object-
>SendQuery("SELECT COUNT(*) FROM `".DBPREFIX."templates` WHERE
`publish`='1';"));
return $data;
}
function getTemplatesList($condition=null){
$data = $this->db_Object-
>getAssoc('*','templates','','',$condition);
return $data;
}

function getModList($condition=null){
$data = $this->db_Object-
>getAssoc('*','modules','order','ASC',$condition);
return $data;
}

function getPageId($link,$com=null)
{
$url = explode("/",$link);
$curCom = !empty($url[2]) ? $url[1] : 'articles';

if($com)
{
$key = array_search($curCom,$com);
$condition = in_array($curCom,$com) ? " OR `com`='$curCom'" :
'';
}else
$condition = '';

$data = $this->db_Object-
>getAssoc('id','menu_item','','',"`link`='$link'".$condition);
return $data[0]['id'];
}

function getRowsCount($table){
```

```
$data = $this->db_Object->getResult($this->db_Object-
>SendQuery("SELECT COUNT(*) FROM `".DBPREFIX.$table."`;"));
return $data;
}
function
updateItem($table,$col,$val,$id,$condition=null,$colName=null)
{
$colName = !empty($colName) ? $colName : 'id';
$condition = !empty($condition) ? ' AND'.$condition : '';

$this->db_Object->SendQuery("UPDATE `".DBPREFIX.$table."` SET
`".$col."`='$val' WHERE `".$colName."`='$id'
".$condition.";");
}

function getCountries(){
$data = $this->db_Object-
>getAssoc('*','country','id_country','ASC');
return $data;
}

function getRegions($id=null){
$condition = !empty($id) ? "`id_country`='$id'" : '';
$data = $this->db_Object-
>getAssoc('*','region','id_region','ASC',$condition);
return $data;
}
function getNext($table){
$data = $this->db_Object->getResult($this->db_Object-
>SendQuery("SELECT MAX(`order`)  FROM `".DBPREFIX.$table."`
;"));
$next = (int)$data + 1;
return $next;
}

function checkUserStatus($id){
$data = $this->db_Object->getResult($this->db_Object-
>SendQuery("SELECT `publish` FROM `".DBPREFIX."users` WHERE
`id`='$id'"));
return $data;
}

function getSettings(){
$data = $this->db_Object->getAssoc('*','settings','','');
return $data[0];
}

function addNewSlug($cols,$values){
$this->db_Object->insert('slugs',$cols,$values);
}

function deleteSlug($com,$action,$param=null){
```

```
if($param)
{
$this->db_Object->SendQuery("DELETE FROM `".DBPREFIX."slugs`
WHERE `com`='$com' AND `action`='$action' AND
`param`='$param'");
}
}

function
getSlug($slug=null,$com=null,$action=null,$param=null){
$rules = array();
if(!empty($slug))
$rules[] = "`slug`='$slug'";
if(!empty($com))
$rules[] = "`com`='$com'";
if(!empty($action))
$rules[] = "`action`='$action'";
if(!empty($param))
$rules[] = "`param`='$param'";

$condition = $rules ? implode(' AND ',$rules) : '';
$data = $this->db_Object-
>getAssoc('*','slugs','','',$condition,1);
return $data[0];
}

function checkDataExisting($table,$id,$colName=null){
$col = $colName ? $colName : 'id';
return $this->db_Object->getResult($this->db_Object-
>SendQuery("SELECT COUNT(*) FROM `".DBPREFIX.$table."` WHERE
`".$col."`='$id';"));
}
}
?>

<-------------------------- View ..... >
<?php
class View
{
// Генерация основного вида для панели администратора
function generate($content_view, $template_view,$com, $data =
null,$prfix = null)
{
include 'templates/'.$template_view;
}

// Подключение вида панели дествий для соответствующего
компонента
function getParts($com,$data = null,$prfix = null){
$file_path = 'components/com_'.$com.'/views/parts.php';
if(file_exists($file_path))
include($file_path);
```

```
}

// Подключение вида впомогательных модальных окон для
соответствующего компонента панели администратора
function getAdditionals($com,$data = null,$prfix=null){
$file_path = 'components/com_'.$com.'/views/additionals.php';
if(file_exists($file_path))
include($file_path);
}

// Подключение любого стороннего вида для соответствующего
компонента панели администратора
function
generateAdditionalView($com=null,$view_name=null,$data =
null,$prfix=null,$file_path=null){
$file_path = empty($file_path) ?
'components/com_'.$com.'/views/'.$view_name : $file_path;
if(file_exists($file_path))
include($file_path);
}

// Подключение любого файла вида для соответствующего модуля
(front-end)
function genModuleView($mod,$view_name,$data = null,$prfix){
$file_path = 'modules/mod_'.$mod.'/views/'.$view_name;
if(file_exists($file_path))
include($file_path);
}

// Генерация основного шаблона для front-end
function genTemplate($template_view,$data=null)
{
include 'templates/'.$template_view.'/index.php';
}
}
?>

<.....Controller.....>
<?php
class Controller {

public $model;
public $view;
public $com;          //Свойство хранит имя компонента
public $admLang;       //Свойство хранит префикс языка панели
администрирования
public $frontLang;   //Свойство хранит префикс языка front-end
protected $url;        //Массив url адресов
protected $slug;       // Алиас страници

function__construct()
{
```

```
$this->model = new Model();
$this->view = new View();
$this->com = strtolower($_GET['com']);
$this->url = explode("/",$_SERVER['REQUEST_URI']);
$this->slug = $this->url{2} ? $this->model->getSlug('',$this-
>url{1},$this->url{2},$this->url{3}) : $this->model-
>getSlug($this->url{1});

$lang = $this->model->getAdmLangList('status',1);
$this->admLang = '_'.$lang['prfix'];
}

function action_index()
{
}

/*
* Отрисовка бокового меню
* панели администратора
*/
function action_getLeftBar($parent,$com_name,$mk=null){
if($menu=$this->model->get_leftBar($parent)){
if($parent != '0' && $menu !== false)
{
echo '<ul class="sub_menu" id="fb' . $parent . '">';

if($mk>1)
echo '<li class="return"
onClick="fast_bar('.$parent.',\'parent\');">
<img src="templates/files/return.png" />
</li>';
}else{
$mk =0;
}

foreach ($menu as $one){

if(strtolower($one['com'])==$com_name && !empty($com_name) &&
$parent==0 && $one['id']!=5)
echo '<li class="active">';
elseif($one['id']=='5')
{
$mass=$this->model->get_leftBar($one['id']);
$fl = 0;
foreach($mass as $m)
if($_GET['com']==$m['com'])
$fl =  1;

echo $fl!=0 ? '<li class="active">' : '<li>';
}elseif($parent==5 && $one['com']==$_GET['com']){
echo '<li class="active">';
}else{
```

```
$curp = $this->model->getLBParent($_GET['com_action']);

echo $one['id']==$curp['parent'] ||$one['id']==$curp['id'] ?
'<li class="active">' : '<li>';
}


$try=$this->model->get_leftBar($one['id']);
$img = !empty($one['img']) ? ' style="background-
image:url('.$one['img'].');"' : '';
if($try) {
$class = $mk!=0 ? 'class="hvr-icon-forward"' : '';

echo '<a  onClick="fast_bar(' . $one['id'] . ');"
'.$class.$img.' title="' . $one['name' . ADMLANG] . '"><span>'
. $one['name' . ADMLANG] . '</span></a>';
}elseif(empty($one['link']))
echo '<a
href="index.php?com='.$one['com'].'&com_action='.$one['action'
].'" '.$img.' title="' . $one['name' . ADMLANG] .
'"><span>'.$one['name'.ADMLANG].'</span></a>';

$this->action_getLeftBar($one['id'],$com_name,$mk+1);
echo '</li>';
}
if($parent != '0' && $menu !== false) echo '</ul>';
}
}


/*
* Вывод насвания компонента и
* текущего действия впанели администратора
*/
function action_getTitle($com){
$title = $this->model->get_Title($com,$_GET['com_action']);
echo '<i class="h3_'.strtolower($com).'"></i>';
if($com!='settings' && $com!='quick' && $com!='template' &&
$com!='customization')
echo '<span>'.$title[0]['name'.ADMLANG].':
<em>'.$title[1]['name'.ADMLANG].'</em></span>';
else
echo '<span>'.$title[0]['name'.ADMLANG].'</span>';

}


/*
* Вывод панели дествий
* для текущего компонента
* в панеле администратора
*/
```

```
function action_getParts($com){
$this->view->getParts($com);
}

/*
* Вывод вспомогательных
* всплывающих окон
*/
function action_getAdditionals($com){
$data = self::action_getLanguagesList("`publish`='1'");
$this->view->getAdditionals($this->com,$data);
}


/*
* Получение основного языка
* отображения front-end
*/
function action_getMainLang(){
$prfix = $this->model->getMainLang();
return '_'.$prfix;
}


/*
* Получение списка языковых
* префиксов для front-end
*/
function action_getPrefixList($condition=null){
$data = $this->model->getPrefixList($condition);
return $data;
}

/*
* Получение списка языков front-end
*/
function action_getLanguagesList($condition=null){
$data = $this->model->getLanguagesList($condition);
return $data;
}


/*
* Проверка уровня доступа
* к компоненту в панели администратора
*/
function action_checkAccessLevels($group,$condition=null){
//Получаем список доступных компонентов и действий для
пользовательской группы $group в панели администратора
$data = $this->model->getGroupAccessLevels($group);
$action = $condition!='addAlbum' ? $_GET['com_action'] :
'createAlbum';
```

```
//Получаем идентификатор текущего действия в компоненте
$action_ids = $this->model-
>getActionId($_GET['com'],$action,$condition);

//Если получен список доступных группе компонентов и действий,
// а так же в БД найден идентификатор текущего действия
if($data && $action_ids)
{
//Если текущее действие не входит в список разрешенных для
группы
//выполнение процесса остановки и возврата к месту начала
действий
if(!in_array($action_ids[0],$data)){
if($_GET['com']!='Ajax')
{
setcookie("error[".time()."]",$GLOBALS-
>translation158,time()+3600,"/");
header("location: " . $_SERVER['HTTP_REFERER'] . "");
exit();
}else{
if($_GET['com_action']=='updateItem')
$txt = $this->model-
>getColItem($_GET['table'],$_GET['column'],$_GET['id']);
else
$txt = $_GET['id'];

echo $GLOBALS->translation158.'||'.$txt;
exit();
}

}
}else{
if($_GET['com']!='Ajax')
{
setcookie("error[".time()."]",$GLOBALS-
>translation158,time()+3600,"/");
header("location: " . $_SERVER['HTTP_REFERER'] . "");
exit();
}else{
if($_GET['com_action']=='updateItem')
$txt = $this->model-
>getColItem($_GET['table'],$_GET['column'],$_GET['id']);
else
$txt = $_GET['id'];

echo $GLOBALS->translation158.'||'.$txt;
exit();
}
}
}
```

```
/*
* Получение флага видимости модуля
* для текущей страници
*/
function action_checkModuleVisibility($visibility,$menu){

if($visibility==0) // Виден на всех страницах
{
return true;
}elseif($visibility==1){ // Ни на одной странице
return false;
}else{
$ids = json_decode($menu['menus'],true);
$com = json_decode($menu['com'],true);
$id = $this->model->getPageId($_SERVER['REQUEST_URI'],$com);
//Получаем идентификатор пункта меню по ссылке в адресной
строке

if($visibility==2)
{
$fl = in_array($id,$ids) ? true : false;       // Виден только
на указанных страницах
}else{
$fl = !in_array($id,$ids) ? true : false;      // Виден на всех
кроме указанных
}

return $fl;
}
}

/*
* Получение количества
* записей в таблице
*/
function action_getRowsCount($table){
$data = $this->model->getRowsCount($table);
return $data;
}

function action_getCountries(){
$data = $this->model->getCountries();
return $data;
}

function action_getRegions(){
$data = $this->model->getRegions();
return $data;
}

function checkUserStatus($id){
$data = $this->model->checkUserStatus($id);
```

```
return $data;
}


function createSlug($slug,$com,$action,$param){
$cols = array(``id``,``slug``,``com``,``action``,``param``);
$values = array(
"'NULL'",
"'$slug'",
"'$com'",
"'$action'",
"'$param'"
);

$this->model->deleteSlug($com,$action,$param);
$this->model->addNewSlug($cols,$values);
}

function getSlugParam(){
if(empty($this->url{3}))
{
$id = $this->slug['param'];
}else{
$id = $this->url{3};
}
return $id;
}

function checkDataExisting($table,$id,$colName=null){
$row = $this->model->checkDataExisting($table,$id,$colName);
if($row==0)
Route::ErrorPage404();
}
}
?>
```

Принцип роботи та рейтинг популярності CMS систем

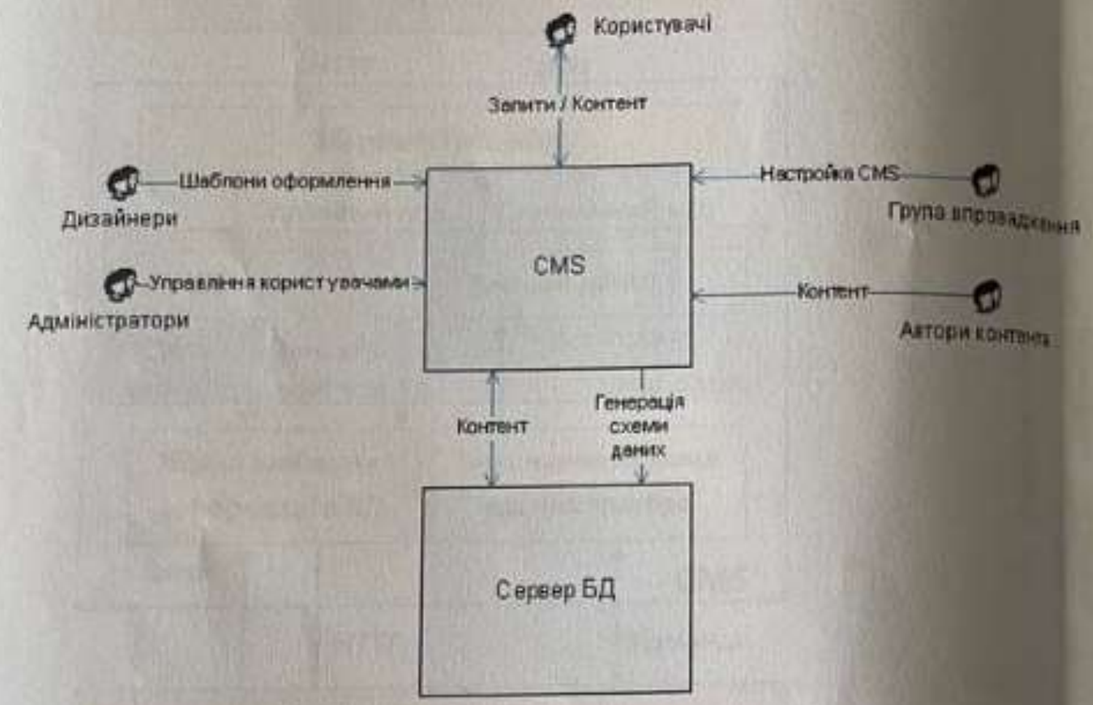


Рисунок 1 – Схема функціонування CMS

Рисунок 2 – Рейтинг популярності безкоштовних CMS

					13.02070849.00046 Пл1	
Зм.	Лист	№ докум.	Підп.	Дата		
Розроб.		Юрчева Д.С.			Розробка системи управління веб-контентом Принцип роботи та рейтинг популярності CMS систем	Літ. / Маса / Масшт.
Перев.		Касьян М.М.				Аркуш 1 / Аркушів 1
Т.контр.						
Н.контр.		Щербак Н.В.				НУЗП КНТ-527
Затв.		Кудрявцев Р.К.				

Схема функціонування CMS

Рисунок 1 – Схема функціонування CMS

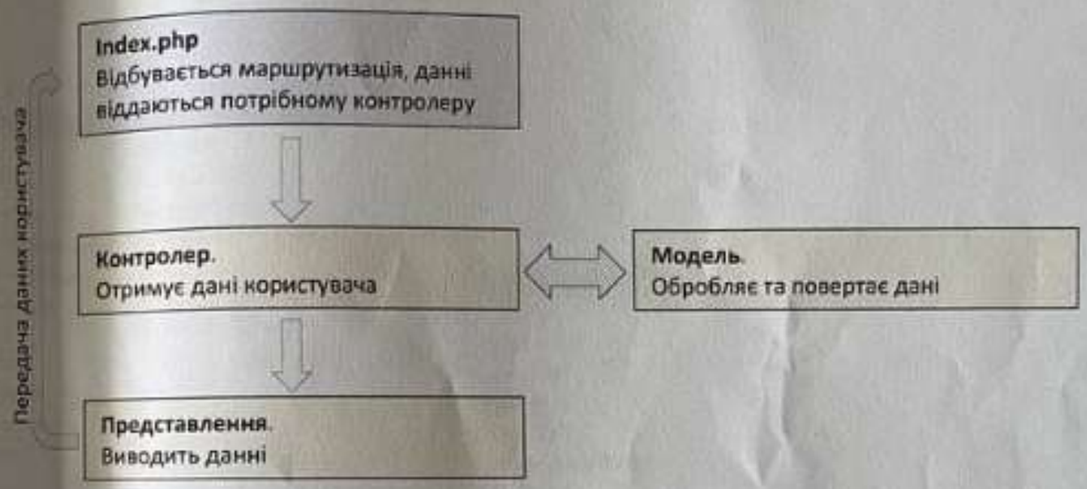
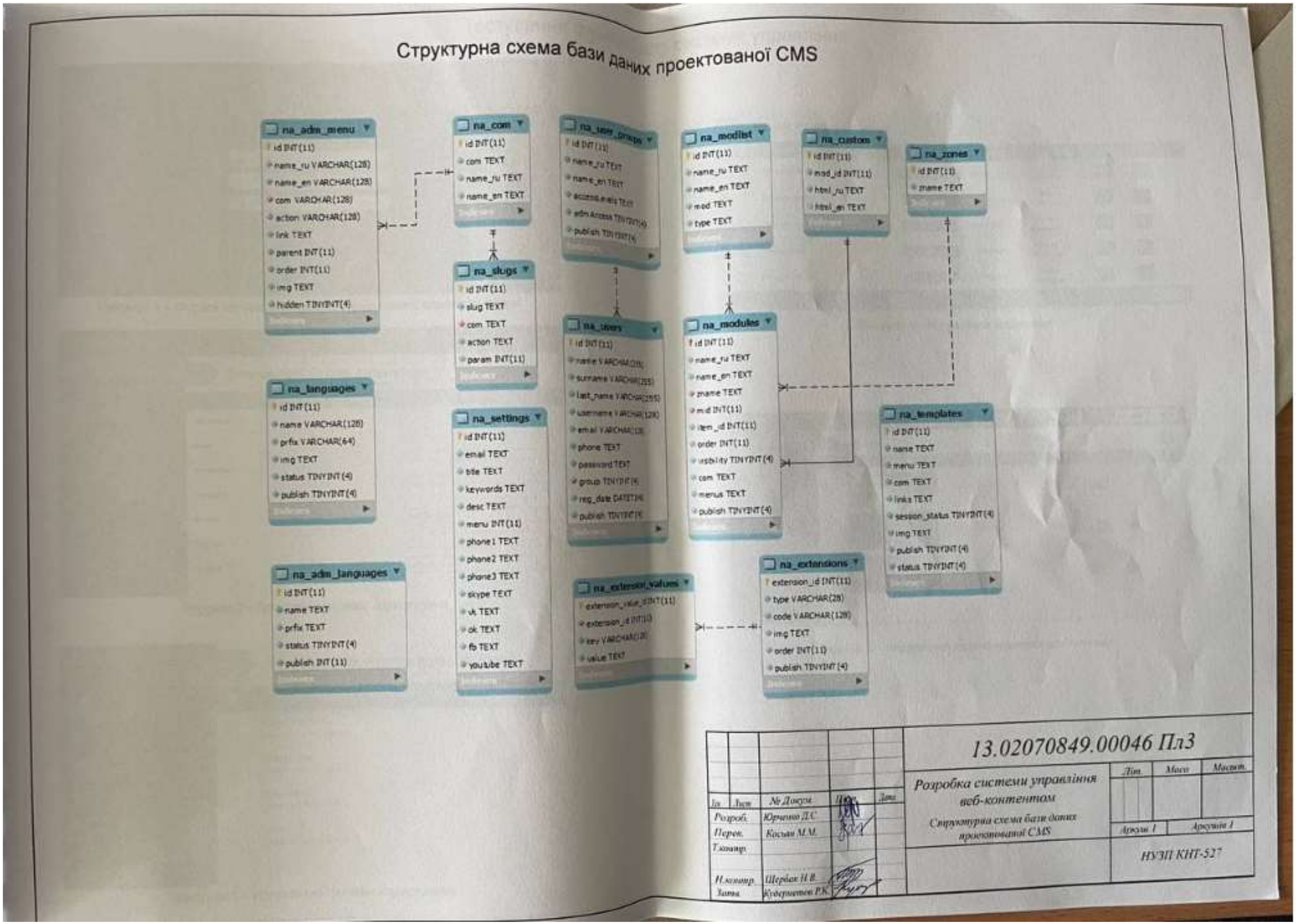


Рисунок 2 – Схема функціонування ядра CMS побудованої на базі шаблона MVC

					13.02070849.00046 Пл2	
Зм.	Арк.	№ докум.	Підп.	Дата	Розробка системи управління веб-контентом Схеми функціонування CMS	Літ. / Маса / Масшт.
Розроб.		Юрченко Д.С.				
Перев.		Касьян М.М.				Аркуш 1 / Аркушів 1
Т.контр.						
Н.контр.		Щербак Н.В.				НУЗП КНТ-527
Затв.		Кудрявцева Р.К.				

Структурна схема бази даних проектованої CMS



					13.02070849.00046 Пл3	
Зм.	Лист	№ докум.	Підп.	Дата	Розробка системи управління веб-контентом	Літ. / Маса / Масшт.
Розроб.		Юрченко Д.С.			Структурна схема бази даних проектованої CMS	Аркуш 1 / Аркушів 1
Перев.		Косьян М.М.				НУЗП КНТ-527
Т.контр.						
Н.контр.		Щербак Н.В.				
Затв.		Кудерметов Р.К.				

Рисунок 1 – Форма авторизації для входу до панелі адміністрування

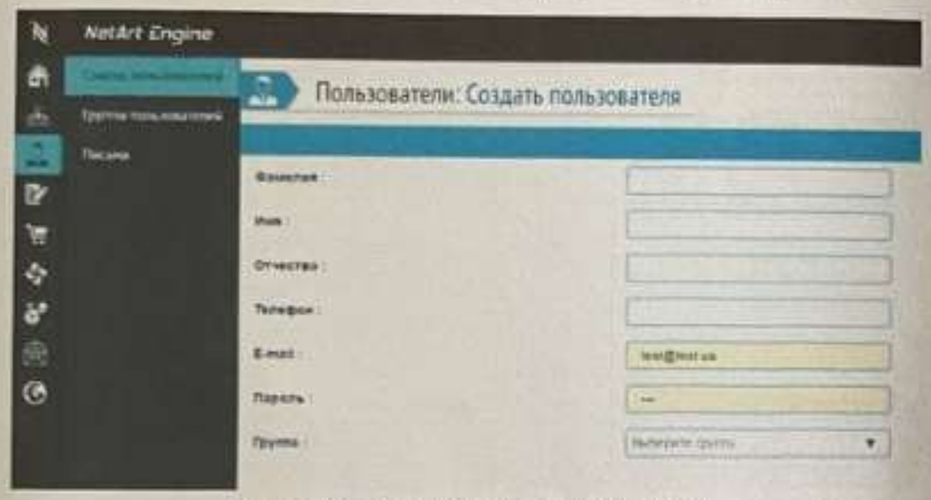


Рисунок 2 - Додавання нового користувача

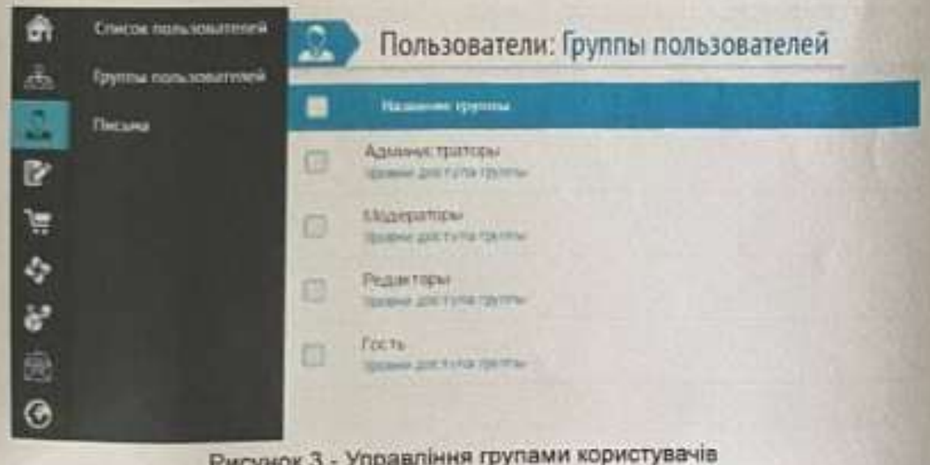


Рисунок 3 - Управління групами користувачів

Рисунок 4 - Управління модулями

Рисунок 5 - Управління встановленими шаблонами

					13.02070849.00046 Пл4		
Зм.	Арк.	№ докум.	Підп.	Дата	Розробка системи управління веб-контентом	Літ.	Маса	Масшт.
Розроб.		Юрченко Д.С.			Тестування функціоналу системи управління			
Перев.		Касьян М.М.				Аркуш 1	Аркушів 1	
Т.контр.						НУЗП КНТ-527		
Н.контр.		Шерстюк Н.В.						
Затв.		Кудрявцев Р.В.						

Рисунок 1 – Результати тестування для мобільних пристроїв

Рисунок 2 - Результати тестування для комп'ютерів

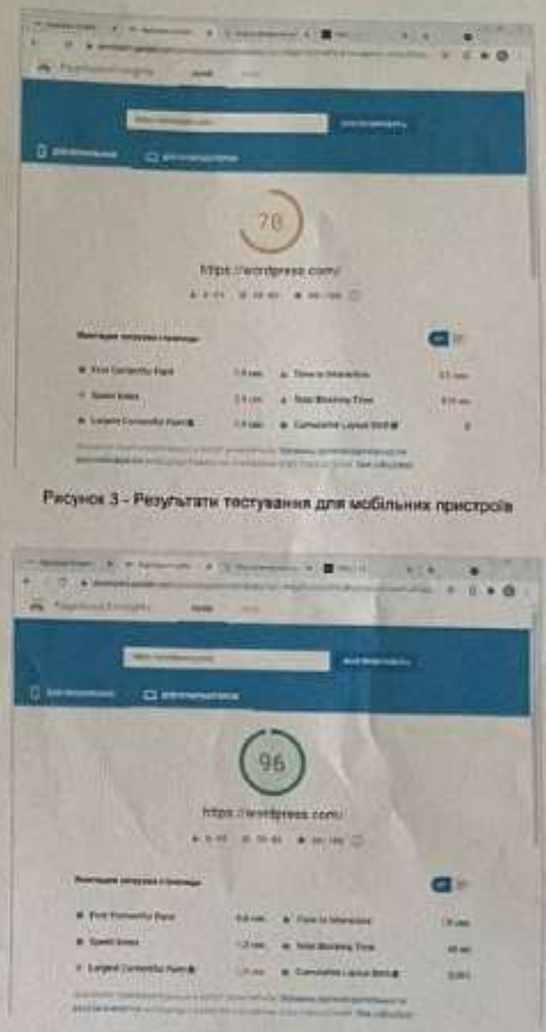


Рисунок 3 - Результати тестування для мобільних пристроїв

Рисунок 4 - Результати тестування для комп'ютерів

					13.02070849.00046 Пл5	
Змн.	Лист	№ докум.	Підп.	Дата	Розробка системи управління веб-контентом Тестування продуктивності розробленої системи в Google Lighthouse	Літ. / Маса / Масшт.
Розроб.		Юрчишин Д.С.				
Перев.		Касьян М.М.				Аркуш 1 / Аркушів 1
Т.контр.						
Н.контр.		Щербак Н.В.				НУЗП КНТ-527
Затв.		Жуковський Р.К.				