МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ КОРИСТУВАЧА ON-LINE ВИДАННЯ»**

Виконала: студентка 2 курсу, групи 8.1228
спеціальності 122 комп'ютерні науки
освітньої програми комп'ютерні науки
(шифр і назва спеціальності)
І.О. Бондаренко
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н., Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н., Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122комп'ютерні науки

(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерних наук,к.т.н.,
доцент

_____ Борю С.Ю.
(підпис)

« 30 » травня 2019 р.

ЗАВДАННЯНА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Бондаренко Ірині Олександрівні
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка програмного забезпечення для автоматизації роботи користувача on-line видання

керівник роботи Матвіїшина Надія Вікторівна, к.т.н, доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи 16.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3.Розроблений програмний продукт згідно з темою диплому

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 30.05.2019

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	03.09.2019	
2.	Збір вихідних даних.	10.09.2019	
3.	Обробка методичних та теоретичних	20.09.2019	
	джерел.		
4.	Розробка першого тадругого розділу.	28.09.2019	
5.	Розробка третього розділу.	20.10.2019	
6.	Оформлення та нормоконтроль	16.12.2019	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	14.01.2020	

Студент _____ І.О. Бондаренко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного забезпечення для автоматизації роботи користувача on-line видання»: 54 с., 34 рис., 1 табл.,17 джерел, 12 додатків.

АВТОМАТИЗАЦІЯ, АТРИБУТ, БАЗА ДАНИХ, ЗАПИТ, ЗВ'ЯЗОК, ІНТЕРНЕТ-ВИДАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, JAVASCRIPT, MYSQL, PHP, SEO-ОПТИМІЗАЦІЯ,WEB-ДОДАТОК.

Об’єкт дослідження – періодичні он-лайн видання.

Мета роботи: проектування програмного забезпечення для автоматизації роботи користувачів on-line видання та його реалізація.

Метод дослідження – аналітичний, оглядовий.

Задачі кваліфікаційної роботи:

− розглянути принципи автоматизації та оптимізації роботи користувачів інформаційних систем;

− розглянути технології та засоби розробки програмного забезпечення для автоматизації роботи користувачів інформаційної системи;

− розробка програмного забезпечення для автоматизації (спрощення та прискорення роботи користувача) та оптимізації роботи онлайн видання.

Результатом роботи є додаткове програмне забезпечення, що автоматизує роботу користувача щоденного он-лайн видання (інтернет-журналу).

SUMMARY

Master's Qualification Thesis «Software Development for the Automatization of the User’s On-line Edition Work»: 54p., 34 figures, 1 table, 17references,12 appendixes.

ATTRIBUTE, AUTOMATION, DATABASE, INFORMATION SYSTEM, INTERNET-EDITION,JAVASCRIPT, MYSQL, PHP, RELATION, REQUEST, SEO-OPTIMIZATION, WEB-APPLICATION.

The object of the study is periodical online edition.

The aim of the study is designing and implementing a daily on-line edition with the ability to administer, write, edit and comment on articles.

Designingand implementing software to automate the online edition users’ work.

The methods of research are analytical, overview.

Tasks of the master’s qualification work:

− to consider principles of automation and optimization of information systemusers’ work;

− to consider technologies and tools for creating software for automation information system users’ work;

− to consider software development issues;

− software development for automation (simplification and acceleration of users’ work), and optimization of online magazine work.

The result of the work is an additional software that allows to automates the work of daily online edition (online magazine)users’.

ЗМІСТ

ВСТУП

Інтернет-журнал – періодичне видання у мережі Інтернет, яке може існувати як незалежне видання, так і як онлайн-версія друкованого журналу.

Такі інтернет-журнали набувають попиту на сьогоднішній день у зв’язку з простотою використання. Інтрнет-журнали можуть буди з платною підпискою, але більшість з них мають безкоштовну основу. Головною перевагою саме таких видань є мобільність та швидкий доступ до потрібної інформації, потрібен лише гаджет (ноутбук, комп’ютер або планшет) та доступ до Інтернету.

Раніше було розроблено додаток, який має типову структуру онлайн видання та дозволяє читачу приєднатися до команди авторів. Спираючись на досвід роботи з цим додатком, було вирішено покращити роботу періодичного онлайн видання. Запропонований інтернет-ресурс має приємний веб-інтерфейс та є простим у використанні. На сам перед, метою роботи є вдосконалення вже існуючих функцій. Під цим можна розуміти:

− оновлення кабінету користувача;

− додавання функцій для поліпшення роботи авторів, редакторів та модераторів сайту;

− структурована сторінка з навчальними відео-роліками для користувачів;

− автоматизація завантаження статей на сайт.

Головною метою роботи, що розглядається, є оптимізація періодичного онлайн видання за допомогою додаткового програмного забезпечення для спрощення та покращення роботи користувачів з додатком.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

Періодичне видання – це видання, зазвичай друковане, яке публікується постійно та з певним проміжком часу. Можна виділити різні типи видань за часом публікації. Наприклад, щоденні, щотижневі, щомісячні, щорічні та інші.

У цій роботі ми детально зупинимося на процесі автоматизації та оптимізації роботи онлайн видання.

На сьогоднішній день Інтернет набув широкого поширення серед населення, в зв'язку з чим для роботи і пошуку інформації зручніше використовувати свій персональний гаджет. Для користувача важлива швидкість і зручність при користуванні Інтернет-ресурсами, в тому числі і періодичними онлайн виданнями.

Раніше було розроблено додаток, який має типову структуру онлайн видання та дозволяє читачу приєднатися до команди авторів. Спираючись на досвід роботи з цим додатком, було вирішено покращити роботу періодичного онлайн видання. Запропонований інтернет-ресурс має приємний веб-інтерфейс та є простим у використанні. На сам перед, метою роботи є вдосконалення вже існуючих функцій та загальна оптимізація роботи ресурсу.

Під цим мається на увазі розробка додаткового програмного забезпечення на основі проведених досліджень. Для початку, розглянемо інтернет видання які існують на окремих веб-сторінках та не мають друкованого примірника. Далі розглянемо засоби які використовують для автоматизації та оптимізації веб-ресурсів.

1.1.1 Веб-ресурс brodude.ru

Щоденний чоловічий журнал. Приємне оформлення, але незручне розміщення статей (рис. 1.1). Додаток має можливість зареєструватись через різні соціальні мережі, пошук по статтям, коментування статей.

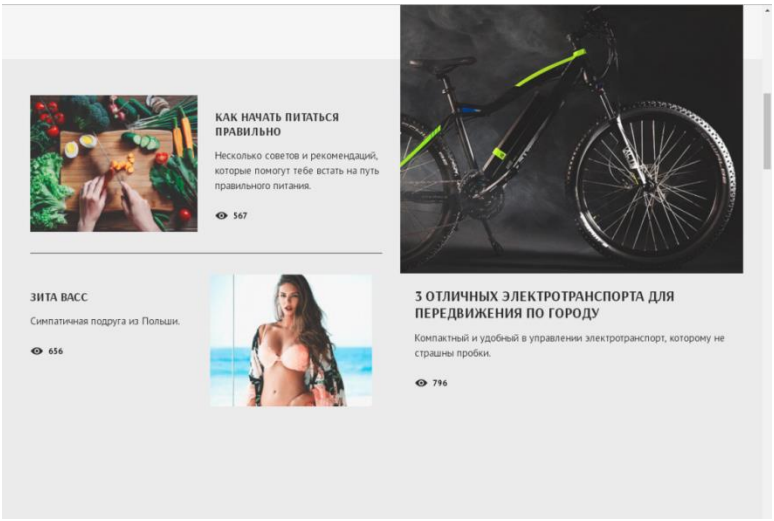


Рисунок 1.1 – Розміщення статей

1.1.2 Веб-ресурс im-ho.media

Інтернет видання має приємне оформлення, форму реєстрації, пошук по сайту (рис. 1.2).

Рисунок 1.2 – Головна сторінка

Усі випуски за інші роки знаходяться в архіві (рис. 1.3).

Рисунок 1.3 – Архів

Випуск видання має власну обкладинку, сторінку, на якій розміщені посилання на статті випуску та PDF-версію для завантаження (рис. 1.4).

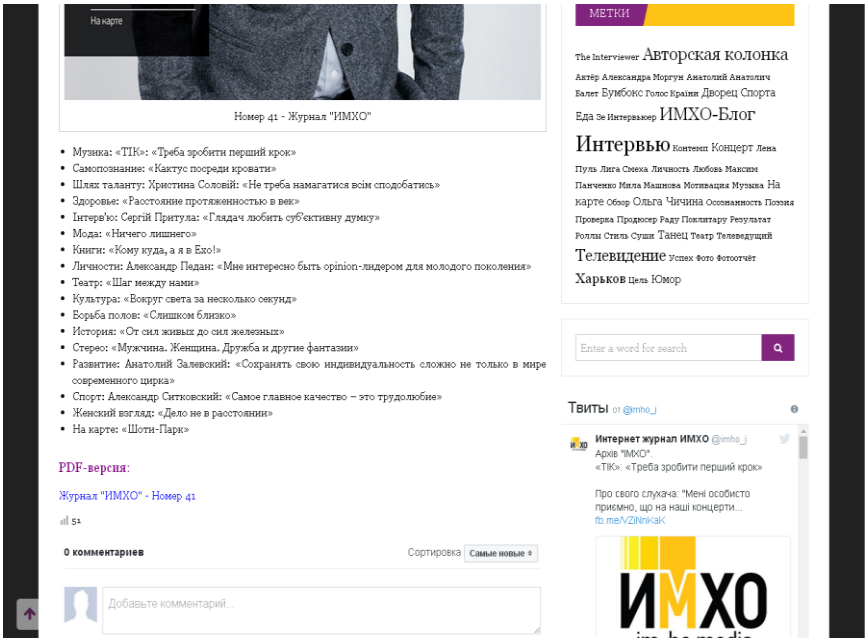


Рисунок 1.4 – Сторінка з випуском

1.1.3 Веб-ресурс fainaidea.com

Приємне тематичне оформлення, пошук по статтям, можливість оформити підписку на e-mail адресу (рис. 1.5, 1.6).

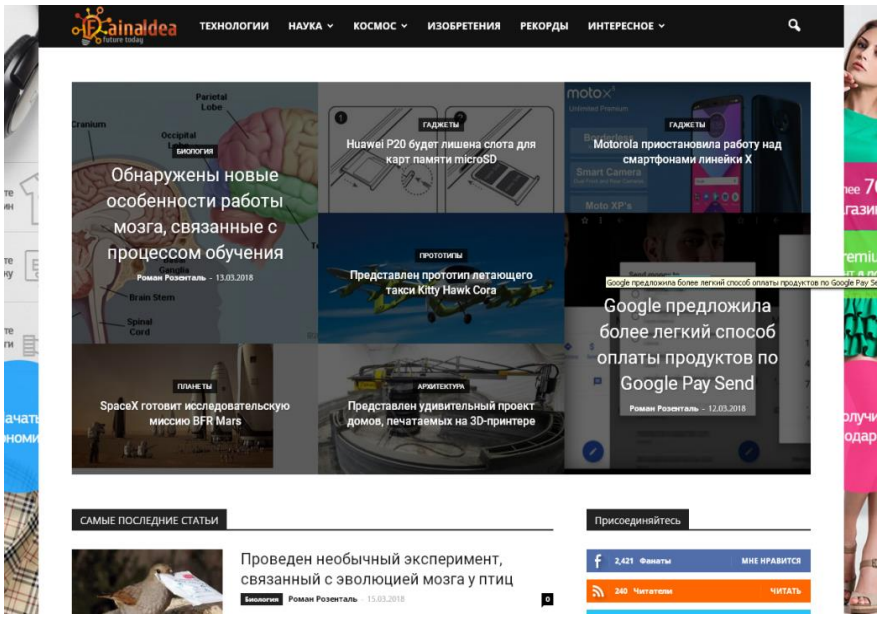


Рисунок 1.5 – Головна сторінка

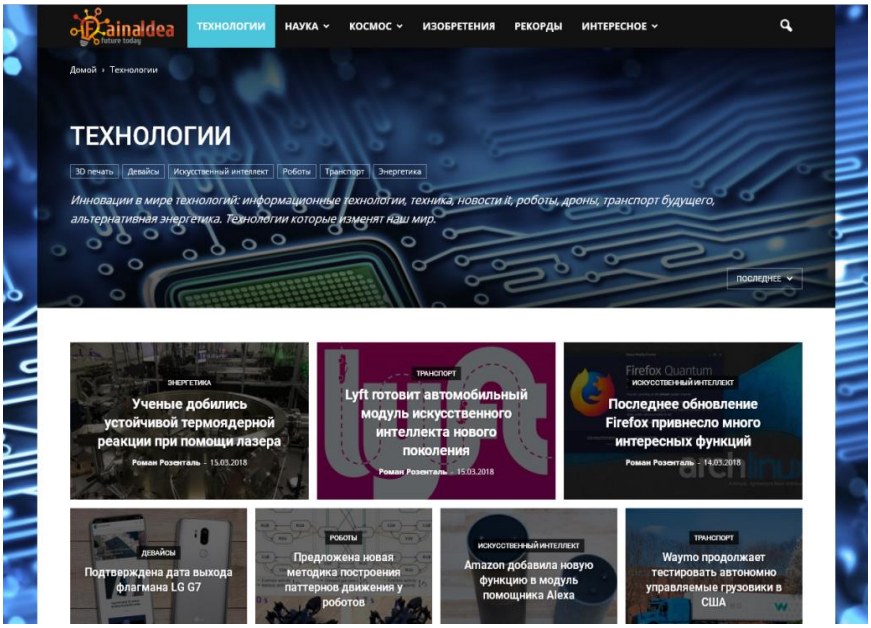


Рисунок 1.6 – Список статей

1.2 Засоби оптимізації веб-додатків

У кожній професійній області та сфері діяльності присутня конкуренція. Це можна сказати і про Інтернет індустрію, яка сьогодні є одним з лідируючих методів просування будь-якого бізнесу в мережі Інтернет або побудова бізнесу безпосередньо в Інтернеті. Наприклад:

- розміщення реклами на власному сайті;
- платні функції веб-ресурсу;
- продаж товарів або послуг за допомогою веб-сайту;
- і так далі.

Одним з найбільш ефективних і дієвих інструментів для просування сайту є його оптимізація [10].

Інтернет простором, як рекламним майданчиком, стали котируватися останнім часом досить активно. Адже зі збільшенням числа Інтернет користувачів в прямій геометричній прогресії починає рости і кількість потенційних клієнтів бізнесу будь-якої сфери. Тому, грамотні власники компаній звернули увагу на рекламу в Інтернеті, і стали активно використовувати її в своїх рекламних кампаніях. Отже, виникла потреба і в оптимізації сайтів, для того щоб ті займали високі місця у видачі пошукових систем, за допомогою яких потенційний клієнт може шукати інформацію в Інтернет просторі.

Оптимізація також актуальна для сайтів, які не мають на увазі собою просування бізнесу.

Наприклад, різні соціальні мережі. Спочатку соціальні мережі не мали на увазі під собою наявність платних послуг, але з ходом часу такі послуги з’явилися. Поява такого роду послуг не заважає звичайним користувачам, але може допомогти більш просунутим.

Онлайн видання можуть отримувати прибуток, як і на продажу платних функцій, так і на надання рекламного майданчика. Також, з часом і набуттям популярності, такі веб-сайти можуть зробити свій журнал платним. Це стане

причиною зменшення читачів, бо в першу чергу такі ресурси цінуються за доступність. Слід провести аналіз і зайнятися оптимізацією веб-ресурсу перш ніж думати про будь-який прибуток з власного веб-сайту.

Є декілька варіантів оптимізації вже готового веб-ресурсу. Наприклад, можна скористатися послугами дослідної веб-студії, де оптимізацією займається спеціально навчена людина.

У нашому випадку, ми самі будемо працювати над оптимізацією роботи веб-сайту. Оптимізацію поділяють на зовнішню та внутрішню.

1.2.1 Зовнішня оптимізація

Зовнішня оптимізація включає в себе роботу над рекламою веб-сайту за допомогою зовнішнього просування. Вона може включати в себе наступні моменти:

- закупівлю посилань і статей на схожих за тематикою ресурсах;
- контекстну рекламу і банерне просування;
- розкрутку на тематичних форумах і в блогах.

1.2.2 Внутрішня оптимізація

Внутрішня оптимізація веб-сайту є найбільш важливим моментом для гарного рейтингу ресурсу. Такий метод оптимізації включає в себе складання семантичного ядра, написання унікального СЕО-контенту, перелінковку сторінок всередині ресурсу, поліпшення функціоналу веб-ресурсу.

Семантичне ядро сайту (СЯ) – це упорядкований набір слів, їх морфологічних форм і словосполучень, які найбільш точно характеризують вид діяльності, товари або послуги, пропоновані сайтом.

Складаючи смислове ядро, ви відповідаєте на глобальне питання: яку інформацію можна знайти на сайті. На створення семантичного ядра можна дивитися з іншого боку, оскільки одним з головних принципів бізнесу і

маркетингу вважається клієнтоорієнтованість. Потрібно визначити, за допомогою яких пошукових запитів користувачі шукають інформацію, яка буде опублікована на сайті.

Побудова смислового ядра вирішує ще одну задачу. Наразі йдеться про розподіл пошукових фраз по сторінках ресурсу. Працюючи з ядром, ми визначаємо, яка сторінка найточніше відповідає на конкретний пошуковий запит або групу запитів.

Є два підходи до вирішення цього завдання. Перший передбачає створення структури сайту за результатами аналізу пошукових запитів користувача. В цьому випадку семантичне ядро визначає каркас і архітектуру ресурсу. Другий підхід передбачає попереднє планування структури ресурсу до аналізу пошукових запитів. В цьому випадку семантичне ядро розподіляється по готовому каркасу.

Обидва підходи так чи інакше працюють. Але буде логічно спочатку планувати структуру сайту, а потім визначати запити, за якими користувачі зможуть знайти ту чи іншу сторінку. В цьому випадку ми залишаємося проактивними, тобто самі обираємо, що треба розповідати потенційним клієнтам. Якщо ми підганяємо структуру ресурсу під ключі, то залишаємося об’єктом і реагуємо на середу, а не активно її змінюємо.

Запланований результат побудови семантичного ядра – це список ключових запитів, розподілених по сторінках сайту. Він містить URL сторінок, пошукові запити і вказівку їх частотності.

Для того щоб підібрати семантичне ядро, потрібно розуміти, що таке ключові слова і які ключі використовує аудиторія. З цими знаннями можна коректно використовувати один з інструментів для підбору ключових слів.

Ключі – це слова або фрази, які використовують потенційні клієнти, щоб знайти необхідну інформацію. Наприклад, щоб приготувати торт, користувач вводить в пошуковий рядок запит «наполеон рецепт з фото».

Ключові слова класифікуються за кількома ознаками. За популярністю виділяють високо-, середньо-і низькочастотні запити. За різними даними, пошукові фрази об’єднуються в групи так:

− до низькочастотних відносяться запити з частотою показів до 100 в місяць. Деякі фахівці включають в групу запити з частотою до 1000 показів;

− до середньочастотних відносяться запити з частотою до 1000 показів. Іноді експерти збільшують поріг до 5000 показів;

− до високочастотним запитам відносяться фрази з частотою від 1000 показів. Деякі автори вважають високочастотними ключі, що мають від 5000 або навіть 10 000 запитів.

Існує багато інструментів для підбору ключових слів. Можна побудувати ядро за допомогою платних або безкоштовних сервісів і програм. Слід вибирати конкретний засіб в залежності від поставлених завдань.

Далі переходимо до написання унікального СЕО-контенту. Оптимізація тексту є найпершим етапом у просуванні сайту. Її можна проводити разом з написанням статті, або ж безпосередньо після. Рекомендується використовувати перший варіант, а ще краще - оптимізувати SEO контент ще до його написання. Для початку потрібно скласти докладний план дій, після чого вже приступати до роботи.

Дана схема дозволить уникнути неграмотно вписаних ключових фраз і захистить від втрати логіки в самій статті. Крім того, SEO контент вийде органічним і легким для сприйняття користувачем. Якщо правильно оптимізувати написаний текст, то позиції в пошуковій видачі будуть дуже високими.

Написання грамотного і красивого тексту – завдання не з простих. Досить складно дотримати в статті необхідну кількість ключових запитів, не втративши при цьому її логіки і переданого сенсу.

На щастя, дана сфера діяльності налічує безліч професійних копірайтерів, які неодмінно, за помірну плату, допоможуть з вирішенням цього непростого завдання.

Так як головна особливість програмного продукту в тому, що читач може стати автором статей, то буде кращим проводити оптимізацію статті після написання або ж навчати нових авторів в потрібному напрямку.

Наступним етапом оптимізації може бути внутрішня перелінковка сайту. Внутрішня перелінковка потрібна для двох основних цілей – поліпшити зручність користування сайтом і розподілити вагу ключових слів по сайту для того що б пошукові системи краще індексували сайт і передавали вагу на потрібні сторінки і розділи.

Велику роль відіграє внутрішня навігація. Важливо, щоб будь-яка сторінка на сайті була доступна не більше трьох кліків від головної сторінки для практично всіх видів сайтів, окрім дуже великих сайтів, де нараховуються мільйони сторінок. На таких веб-ресурсах четвертий рівень вкладеності є нормою.

Тобто потрібно зробити так, щоб навігація на сайті була грамотною. Наприклад, зліва можна побачити правильну навігацію, а праворуч некоректну (рис 1.7).

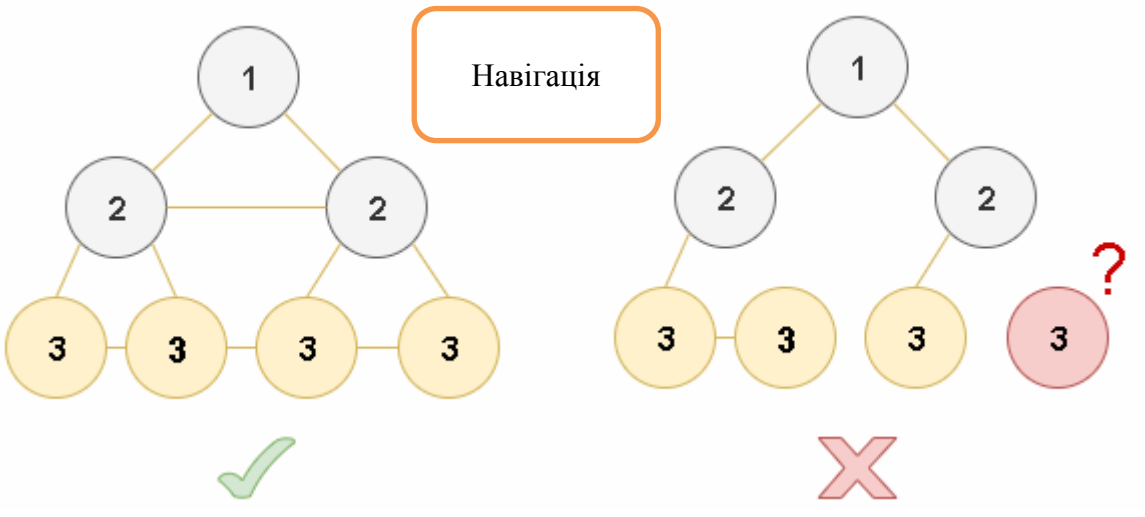


Рисунок 1.7 – Приклад навігації

Для зменшення рівня вкладеності сторінок, використовуємо карти сайтів, хмари тегів, спеціальні скрипти.

Коли в нас є семантичне ядро для сайту (запити розподілені по сторінках) нам необхідно розподілити так само вагу ключових слів по сайту.

Що б розподілити вагу ключових слів можна:

− використовувати спеціальні скрипти або програми;

− зробити це руками (якщо сайт не великий).

Для перелінковки сайту можна використовувати схеми, які розглянемо далі.

Перелінковка зірочкою – перевірений спосіб внутрішньої перелінковки. Основне його завдання зробити так, щоб всі сторінки сайту, посилалися один на одного.

При цьому методі перелінковки всі сторінки мають однакову вагу. Цей метод підходить для сайтів візиток, з невеликою кількістю сторінок.

Перелінковка кільцем – класична схема, при якій сторінки лінкуются кільцем і посилаються на потрібну сторінку. Наприклад, на пріоритетну сторінку.

Так само схему кільце можна поліпшити, і більше ваги передавати на необхідну сторінку сайту. Дана схема перелінковки дає можливість підстрахувати себе, якщо якась сторінка зі схеми вилетить з індексу пошукових систем.

Також можна зробити більш просунуте кільце. При цій схемі утворюється додаткове кільце, яке тісно пов'язане з внутрішнім кільцем. Додаткове кільце виступає страховкою.

Дана схема перелінковки використовується у випадках, коли необхідно передавати більше ваги, тобто задіяти більше сторінок, які будуть посилатися на необхідну нам сторінку.

Перелінковка ієрархічна – схема, яка має за основу вибудовування певної структури, що нагадує прогресію.

Це добра схема внутрішньої перелінковки сайту, яка стабільно працює, перевірена на практиці просування сайтів, але основний її мінус в тому, що

коли одна сторінка із схеми випадає з індексу, то можна отримати помітний мінус в позиціях на сторінках пошуку.

Існують також інші схеми перелиінковки, вони часто складніше і служать для певних задач.

Якщо сайт не об’ємний, наприклад, до п’ятисот сторінок, то внутрішню перелінковку можна зробити ручним методом.

Але коли ми маємо справу зоб’ємними сайтами, то, швидше за все, доведеться використовувати скрипти і програми, які зможуть зробити перелінковку автоматично за заданими критеріями.

Не менш важливим є поліпшення функціональних можливостей веб-сайту. Доробка сайту – це робота по поліпшенню і розвитку Інтернет-ресурсу. Вони потрібні практично завжди, навіть якщо спочатку здається, що це не так. Згодом будь-який ресурс застаріває, з’являються нові тенденції в програмуванні і змінюються алгоритми пошукових систем. Сайт повинен підлаштовуватися під всі зміни.

Перше, на що звертає увагу відвідувач сайту – це дизайн. Роботи з доопрацювання дизайну і оформлення включають в себе:

− зміну дизайну;

− опрацювання логотипу;

− перетворення всіх графічних елементів сайту.

Доробки дизайну потрібні, коли:

− тематика або назва сайту змінилися, що має на увазі інше оформлення або логотип;

− керівництво хоче провести ребрендинг;

− сайт не відповідає тенденціям веб-дизайну.

До поліпшення дизайну потрібно періодично повертатися, тому що будь-який дизайн з часом застаріває.

Наступна частина структури ресурсу – його функціональність, до якої відносяться технічні можливості сайту.

Виділимо в цьому виді доробок кілька напрямків:

- зміна або додавання програмних модулів і віджетів.
- заміна застарілих плагінів;
- розширення версій сайту;
- робота з індексацією.

1.3Автоматизація роботи веб-додатку

Доробки та створення нових функціональних можливостей сайту не менш важливі, ніж робота над візуальною складовою. Це реалізовується за допомогою написання додаткового програмного коду, використовуючи, наприклад, PHP [4], JavaScript [12] або AJAX [14].

JavaScript – це мова керування сценаріями перегляду гіпертекстових сторінок на стороні клієнта. Найбільшу популярність JavaScript забезпечило програмування на стороні клієнта.

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів додатку. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінкам.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам’яттю, прототипне програмування, функції як об'єкти першого класу.

Для мови високого рівня JavaScript має досить сильні можливості: не дозволяє працювати на рівні машиних кодів, однак ви отримуєте доступ до багатьох можливостей браузерів, веб-сторінок, а іноді і системи, в якій працює браузер.Програми на JavaScript обходяться без компіляції, тобто браузеру не доводиться завантажувати віртуальну машину для виконання програмного коду.

Одною з головних причин, по якій веб-розробники взяли JavaScript – можливість виконання на стороні клієнта багатьох функцій, які раніше

виконувалися виключно на стороні сервера, наприклад перевірка форм. У JavaScript елементи форми можна перевірити до того, як користувач передасть інформацію веб-серверу, що призводить до зменшення кількості транзакцій HTTP, а також помітного зниження ймовірності помилки при повторному заповненні форми.

AJAX. Різні технології динамічного підвантаження даних (без перезавантаження сторінки) застосовуються вже досить давно, але поява терміна ajax змусило всіх говорити про неї. На жаль, через відмінності браузерів відсутня єдина кросбраузерна реалізація даної технології.

Технологія AJAX має на увазі використання мови розмітки HTML спільно з таблицями стилів CSS для представлення даних, мови JavaScript і об'єктної моделі документа (Document Object Model, DOM) для маніпуляції даними і мови розмітки XML для обміну інформацією між сервером і клієнтом. Підхід, заснований на базі AJAX, дозволяє підвищити інтенсивність обміну даними між користувачем і серверним додатком, тим самим зменшувати час простою сервера і, що головне, покращувати дружність інтерфейсу.

Завдяки асинхронної взаємодії інтерфейсу з серверної частиною користувачеві відкриваються нові грані зручності.

PHP скриптова мова програмування, створена для генерації HTML-сторінок на веб-сервері та роботи з базами даних. На сьогодні, підтримується більшістю хостингів. Входить до LAMP – «стандартний» набір для створення веб сайтів.

PHP – це оболонка навколо мови C з управлінням пам'яттю та гнучкою системою типів. PHP виконується на веб-сервері, виконується на web-сервере, обробляючи код на вході та генеруючи веб-сторінки на виході. Як і в багатьох інтерпретованих мовах програмування, PHP-скрипти зазвичай зберігаються у вихідних файлах навіть на виробничих веб-серверах.

PHP виконує код, укладений у тег <? php.? > і його підвиди, а інший зміст файлу виводиться одразу на сторінку. Змінні позначаються знаком $ та

не потребують вказівки типу. Ключові слова та синтаксис мови схожі на більшість високорівневих мов програмування.

Використовуючи ці засоби, можливо значно поліпшити роботу користувача за веб-додатком завдяки автоматизації деяких функцій.

1.4 Висновки

Без модернізації сайту і його компонентів портал не зможе залишатися сучасним і затребуваним. Навіть якщо він займає високі сходинки пошукача, при відсутності робіт з модернізації ресурс неминуче почне втрачати позиції.

Як ми розібралися, доопрацювання сайту – необхідний комплекс заходів автоматизації та оптимізації існуючих функцій. Доробками бажано займатися постійно. Високий темп доробок не є обов'язковим, головне – регулярність.

Головним етапом роботи є аналіз існуючих методів для оптимізації та автоматизації роботи раніше розробленого веб-ресурсу. Після ознайомлення з деякими статтями та їх аналізу, було прийнято рішення. Основним руслом в оптимізації вибрати поліпшення функціональних можливостей. Також, важливим, є сео-оптимізація текстів.

Поліпшення функціональних можливостей буде реалізовано шляхом розробки додаткового програмного забезпечення. Перелінковка сайту наразі не потрібна, так як на даний момент навігація на сайті добре організована.

Якщо задуматися про seo-оптимізації тексту, то найоптимальнішим рішенням в цьому проекті є навчання поточних і нових авторів цієї технології. Для цього будуть розміщені статті та відео-уроки.

Головною метою роботи, що розглядається, є оптимізація періодичного онлайн видання за допомогою додаткового програмного забезпечення для спрощення та покращення роботи користувачів з додатком.

2 СТРУКТУРА ВЕБ-ДОДАТКУ ТА БАЗИ ДАНИХ

Представлена робота базується на онлайн виданні, що вже існує. До оптимізованого ресурсу було додано:

– сторінка з відео-уроками для авторів та редакторів;

– сторінка зі статтями для редагування у кабінеті редактора;

– сторінка для публікації статті з файлу.

До можливостей веб-додатку було додано:

– публікація статті із файлу;

– зручний спосіб додавання зображень до статті;

– покращена система редагування статей.

2.1 Основні відомості про PHP, AJAXта JavaScript

У даний час **PHP** підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов, що застосовуються для створення динамічних веб-сайтів.

У області веб-програмування, зокрема, серверної частини, PHP – один з популярних сценарних мов (разом з JSP, Perl, Python і мовами, використовуваними в ASP.NET). Популярність в області побудови веб-сайтів визначається наявністю великого набору вбудованих засобів для розробки веб-додатків.

Основні з них:

– автоматичне вилучення POST і GET-параметрів;

– взаємодія з великою кількістю різних систем управління базами даних (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape і Apache Derby,

Informix, Ovrimos SQL, LotusNotes, DB ++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Інтерфейс PDO);

– автоматизована відправка HTTP-заголовків;

– робота з HTTP-авторизацією;

– робота з cookies і сесіями;

– робота з локальними і віддаленими файлами, сокетами;

– обробка файлів, що завантажуються на сервер;

– робота з XForms.

В даний час PHP використовується сотнями тисяч розробників. Згідно з рейтингом корпорації TIOBE, що базується на даних пошукових систем, в травні 2016 року PHP знаходився на 6-му місці серед мов програмування.

PHP є мовою програмування з динамічною типізацією, що не вимагає вказівки типу при оголошенні змінних, так само як і самого оголошення змінних.

До скалярних типів даних відносяться:

– цілочисельний тип (integer);

– число з плаваючою точкою (float, double);

– логічний тип (boolean);

– строковий тип (string).

До нескалярних типів відносяться:

– масив (array);

– об'єкт (object);

– зовнішній ресурс (resource);

– невизначене значення (null).

До псевдотіпов відносяться:

– mixed будь-який тип;

– number число (integer або float);

– callback (string або анонімна функція);

– void відсутність параметрів.

Для спрощення роботи з PHP існують фреймворки і системи керування вмістом. Найсучасніші і найбільш популярні:

– Phalcon;

– Symfony;

– Laravel;

– Kohana;

– Yii;

– Codeigniter.

AJAX – підхід до побудови призначених для користувача інтерфейсів веб-додатків, що полягає в «фоновому» обміні даними браузера з веб-сервером. В результаті, при оновленні даних веб-сторінка не перезавантажується повністю, і веб-додатки стають швидше і зручніше.

Порівняння класичного підходу та AJAX

У класичній моделі веб-додатків:

− користувач заходить на веб-сторінку і натискає на який-небудь її елемент;

− браузер формує і відправляє запит серверу;

− у відповідь сервер генерує абсолютно нову веб-сторінку і відправляє її браузеру. Після чого браузер повністю перезавантажує всю сторінку.

При використанні AJAX:

− користувач заходить на веб-сторінку і натискає на який-небудь її елемент;

− скрипт (на мові JavaScript) визначає, яка інформація необхідна для оновлення сторінки;

− браузер відправляє відповідний запит на сервер;

− сервер повертає лише ту частину документа, на яку прийшов запит;

− скрипт вносить зміни з урахуванням отриманої інформації (без повного перезавантаження сторінки).

Технологія

AJAX – не самостійна технологія, а концепція використання декількох суміжних технологій. AJAX базується на двох основних принципах:

− використання технології динамічного звернення до сервера «на льоту», без перезавантаження всієї сторінки повністю, наприклад з використанням XMLHttpRequest (основний об'єкт);

− використання DHTML для динамічної зміни змісту сторінки.

Дії з інтерфейсом перетворюються в операції з елементами DOM (англ. Document Object Model), за допомогою яких обробляються дані, доступні користувачеві, в результаті чого уявлення їх змінюється. Тут же проводиться обробка переміщень і клацань мишею, а також натискань клавіш. Каскадні таблиці стилів, або CSS (англ. Cascading Style Sheets), забезпечують узгоджений зовнішній вигляд елементів програми та спрощують звернення до DOM-об'єктів. Об'єкт XMLHttpRequest (або подібні механізми) використовується для асинхронного взаємодії з сервером, обробки запитів користувача і завантаження в процесі роботи необхідних даних.

Переваги

Використання AJAX дозволяє скоротити трафік при роботі з веб-додатком завдяки тому, що замість завантаження всієї сторінки достатньо завантажити тільки змінилася частина або взагалі тільки отримати / передати набір даних в форматі JSON або XML, а потім змінити вміст сторінки за допомогою JavaScript.

При правильній реалізації AJAX дозволяє знизити навантаження на сервер в кілька разів.

Зокрема, всі сторінки сайту найчастіше генеруються за одним шаблоном, включаючи незмінні елементи («шапка», «навігаційна панель», «підвал» і т. Д.), Для генерації яких потрібні звернення до різних файлах, час на обробку скриптів (а іноді і запити до БД) - все це можна опустити, якщо замінити повне завантаження сторінки генерацією і передачею лише змістовної частини.

Оскільки завантаження змінилася частини значно швидше, то користувач бачить результат своїх дій швидше і без мерехтіння сторінки (виникає при повній перезавантаження).

Наприклад, при введенні пошукового запиту в Google виводиться підказка з можливими варіантами запиту. На багатьох сайтах при реєстрації користувач вводить ім'я, і відразу ж бачить, є це ім'я чи ні. AJAX зручний для програмування чатів, адміністративних панелей та інших інструментів, які виводять мінливі з часом дані.

Сторінка не перезавантажується, плеєр продовжує працювати. Тому AJAX цінний на аудіо- та відеохостингу.

Три з цих чотирьох технологій – CSS, DOM і JavaScript – складають DHTML (англ. Dynamic HTML). На думку деяких фахівців,засоби DHTML, що з'явилися в 1997 році, подавали великі надії, але так і не виправдали їх.

Як формат передачі даних можуть використовуватися фрагменти простого тексту, HTML-коду, JSON або XML.

Недоліки

Динамічно створювані сторінки не реєструються браузером в історії відвідування сторінок, тому не працює кнопка «Назад», що надає користувачам можливість повернутися до переглянутих раніше сторінок, але існують скрипти, які можуть вирішити цю проблему.

Інший недолік зміни вмісту сторінки при постійному URL полягає в неможливості збереження закладки на бажаний матеріал. Проблему можна успішно вирішити за допомогою History.pushState.

Динамічно завантажувати вміст недоступний пошуковикам (якщо не перевіряти запит, звичайний він або XMLHttpRequest)

Пошукові машини не можуть виконувати JavaScript, тому розробники повинні подбати про альтернативні способи доступу до вмісту сайту (але на 2019 рік вже не актуально).

Багато сервісів статистики ведуть облік переглядів нових сторінок сайту. Для сайтів, сторінки яких широко використовують AJAX, така статистика втрачає актуальність.

Перерозподіляється логіка обробки даних - відбувається виділення і часткове перенесення на сторону клієнта процесів первинного форматування даних. Це ускладнює контроль цілісності форматів і типів. Кінцевий ефект технології може бути знівельовано необґрунтованим зростанням витрат на кодування і управління проектом, а також ризиком зниження доступності сервісу для кінцевих користувачів.

JavaScript може бути вимкнений з міркувань безпеки. І, звичайно ж, AJAX-сторінки важкодоступні не повнофункціональним браузерам, роботам і веб-архівів.

Існують проблеми з відображенням нестандартних кодувань в деяких сценаріях ajax-скриптів.

Про проблеми AJAX і кирилиці багато сказано в обговореннях в Інтернеті.

Здавалося б, AJAX призначений саме для підвищення швидкості. Але коли AJAX-запитів на одній сторінці багато і, наприклад, по кожному кліку підвантажується список, AJAX-сторінка стає навіть повільніше традиційної.

Якщо зв'язок часто втрачається (через втрату несучої або перевантаження каналу), звичайну сторінку можна як мінімум перезавантажити. AJAX-сторінку (наприклад, з «нескінченною» прокруткою) доводиться перезавантажувати з самого початку і шукати, де зупинився. Паралельна робота – особливість AJAX – тут робить ведмежу послугу, ділячи і без того вузький канал на безліч маленьких з'єднань, і велика ймовірність, що якийсь буде розірвано. Частково вирішується API історії.

JavaScript – мультіпарадігменний мову програмування. Підтримує об'єктно-орієнтована, імперативний і функціональний стилі. Є реалізацією мови ECMAScript (стандарт ECMA-262).

JavaScript зазвичай використовується як вбудований мова для програмного доступу до об'єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінок.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

На JavaScript вплинули багато мов, при розробці була мета зробити мову схожим на Java, але при цьому легким для використання непрограмістів. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в веб-розробці.

JavaScript є об'єктно-орієнтованою мовою, але що використовується в мові прототипування обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, - функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання – що додає мові додаткову гнучкість.

Незважаючи на схожий з Сі синтаксис, JavaScript в порівнянні з мовою Сі має корінні відмінності:

- об'єкти з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

У мові відсутні такі корисні речі, як:

- стандартна бібліотека: зокрема, відсутній інтерфейс програмування додатків по роботі з файловою системою, управління потоками введення-виведення, базових типів для бінарних даних;
- стандартні інтерфейси до веб-серверів і баз даних;

− система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

Елемент script, широко використовуваний для підключення до сторінці JavaScript, має кілька атрибутів:

− необов'язковий атрибут type для вказівки MIME-типу вмісту;

− необов'язковий атрибут src, який приймає в якості значення адреса до файлу зі скриптом;

− необов'язковий атрибут charset, який використовується разом з src для вказівки використовуваного кодування зовнішнього файлу;

− необов'язковий атрибут defer вказує, що отримання скрипта відбувається асинхронно, але виконання слід відкласти до тих пір, поки сторінка не буде завантажена повністю;

− необов'язковий атрибут async вказує, що отримання скрипта відбувається асинхронно, а виконання буде вироблено відразу по завершенні скачування. Черговість виконання скриптів не гарантовано.

JavaScript використовується в клієнтської частини веб-додатків: клієнт-серверних програм, в якому клієнтом є браузер, а сервером - веб-сервер, що мають розподілену між сервером і клієнтом логіку. Обмін інформацією в веб-додатках відбувається по мережі. Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є кросплатформеними сервісами.

JavaScript використовується в AJAX, популярному підході до побудови призначених для користувача інтерфейсів веб-додатків, що полягає в «фоновому» асинхронному обміні даними браузера з веб-сервером. В результаті, при оновленні даних веб-сторінка не перезавантажується повністю і інтерфейс веб-додатки стає швидше, ніж це відбувається при традиційному підході (без застосування AJAX).

JavaScript широко використовується в браузерних операційних системах. Так, наприклад, вихідний код IndraDesktop WebOS на 75% складається з JavaScript, код браузерної операційної системи IntOS – на 70%.

Частка JavaScript у вихідному коді eyeOS – 5%, однак і в рамках цієї операційної системи JavaScript грає важливу роль, беручи участь в візуалізації на клієнті і будучи необхідним механізмом для зв’язку клієнта і сервера.

Призначені для користувача скрипти в браузері – це програми, написані на JavaScript, що виконуються в браузері користувача при завантаженні сторінки. Вони дозволяють автоматично заповнювати форми, переформатувати сторінки, приховувати небажаний вміст і вбудовувати бажане для відображення вміст, змінювати поведінку клієнтської частини веб-додатків, додавати елементи керування на сторінку і т.д.

Для управління призначеними для користувача скриптами в Mozilla Firefox використовується розширення Greasemonkey; Opera і Google Chrome надають засоби підтримки користувальницьких скриптів і можливості для виконання ряду скриптів Greasemonkey.

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.

Крім Java 6, існує ряд платформ, які використовують існуючі движки (інтерпретатори) JavaScript для виконання серверних додатків. (Як правило, мова йде про повторне використання двигунів, раніше створених для виконання коду JavaScript в браузерах WWW.)

Віджет – допоміжна міні-програма, графічний модуль якої розміщується в робочому просторі відповідної батьківської програми (англ.), Що служить для прикраси робочого простору, розваги, вирішення окремих робочих завдань або швидкого отримання інформації з інтернету без допомоги веб-браузера. JavaScript використовується як для реалізації віджетів, так і для реалізації движків віджетів. Зокрема, за допомогою JavaScript реалізовані Apple Dashboard, Microsoft Gadgets, Yahoo! Widgets, Google Gadgets, Klipfolio Dashboard.

Вихідний код та скріншот JavaScript-програми, виконуваної за допомогою Seed.

JavaScript використовується для написання прикладного ПО. Наприклад, 16,4% вихідного коду Mozilla Firefox написано на JavaScript.

Google Chrome OS в якості прикладного ПО використовує веб-додатки.

В оточенні робочого столу GNOME є можливість створювати на JavaScript програми, які оперують з бібліотеками GNOME за допомогою Gjs, Seed.

JavaScript також знаходить застосування в якості скриптового мови доступу до об'єктів додатків. Платформа Mozilla (XUL / Gecko) використовує JavaScript. Серед сторонніх продуктів, наприклад, Java, починаючи з версії 6, містить вбудований інтерпретатор JavaScript на базі Rhino. Сценарії JavaScript підтримуються в таких додатках Adobe, як Adobe Photoshop, Adobe Dreamweaver, Adobe Illustrator і Adobe InDesign.

JavaScript використовується в офісних додатках для автоматизації рутинних дій, написання макросів, організації доступу з боку веб-служб.

JavaScript – один з мов програмування, використовуваних для написання макросів в додатках, що входять до складу OpenOffice.org. В OpenOffice.org інтегрований інтерпретатор JavaScript Rhino. Станом на грудень 2009 року підтримка JavaScript носила обмежений характер. Обмеження, властиві розробці макросів OpenOffice.org на JavaScript:

JavaScript має пропедевтичної цінністю, дозволяючи поєднувати при навчанні інформатики інтенсивну практику програмування і широту використовуваних технологій. Викладання даного мови в школі дозволяє створити базу для вивчення веб-програмування, використовувати на уроках творчі проекти. Відповідний курс дозволяє забезпечити поглиблений рівень вивчення інформатики та його має сенс включати в курси за вибором поглибленого рівня підготовки.

JavaScript – потрібну мову для навчання програмуванню ігор. У порівнянні з альтернативами, він функціонально достатній, простий у

вивченні і в застосуванні, знижує складність для навчання, мотивує учнів ділитися своїми іграми з іншими.

У деяких мовах програмування існують засоби підтримки взаємодії з JavaScript-кодом.

Для PHP є пакет HTML Javascript, що надає інтерфейс створення простих JavaScript-програм.

Відповідний пакет для Tcl називається :: javascript. Він надає команди для створення коду HTML і JavaScript.

Пакет для Perl Data :: JavaScript дозволяє переносити структури даних Perl в JavaScript-код.

2.2 Специфікація предметної області

Предметна область та її опис

Предметною областю у даній роботі є періодичне онлайн видання, з можливістю адміністрування, написанням статей та можливістю стати автором статей.

Користувачі

Користувачів можна розділити на наступні групи:

– гості;

– підписчики;

– редактори;

– адміністратори;

– автори;

– модератори.

Гості

Гостями є користувачі: які не пройшли авторизацію. Вони мають доступ до перегляду статей і коментарів, перегляду архіву, популярних

записів, пошуку по заголовкам статей. Вони не мають права коментувати запис і виконувати будь-які дії з об'єктами сайту або користувачами.

Підписчики

Підписчиками називаються користувачі, які зареєстровані на сайті і пройшли авторизацію на сайті. Вони мають ті ж можливості, що і гості, за вийнятком того, що підписчики мають право коментувати записи, але точно так само не мають право на виконання будь-яких дій зі статтями, коментарями, користувачами.

Редактори

Редакторами є користувачі з додатковими правами на додавання і редагування статей, публікацію або видалення коментарів, перегляд всіх користувачів і їх ролей.

Адміністратори

Адміністраторами сайту називаються користувачі, які мають доступ до зміни, додавання, видалення будь-яких об'єктів сайту: статті, коментарі, призначення прав користувачам.

Автори

Авторами сайту називаються користувачі, які мають доступ до додавання і зміни власних статей.

Модератори

Модераторами сайту називають користувачів, які мають повний доступ до коментарів і приймають рішення про публікацію коментарів.

Визначення основних операцій над даними:

– реєстрація;

– авторизація;

– вихід з облікового запису;

– додавання статті;

– додавання коментаря;

– редагування статті;

– редагування прав користувача;

– видалення статті;

– видалення коментаря;

– видалення категорії;

– видалення користувача;

– перегляд однієї статті;

– перегляд статей за критерієм;

– заявка на зміну ролі.

Елементи даних, які використовуються (табл. 2.1).

Таблиця 2.1 – Визначення використаних елементів даних

Назва	**Визначення**
Користувач	
Номер користувача	Однозначно визначає кожного зареєстрованого користувача
Логін	Визначає ім'я кожного користувача
Пароль	Визначає пароль для кожного користувача
E-mail	Визначає email користувача
Роль	Визначає, до якої групи належить користувач
Коли створено	Визначає дату створення облікового запису
Коли оновлений	Визначає останній вхід користувача
Стаття	
Номер статті	Однозначно визначає номер кожного створеного поста
Заголовок	Визначає назву статті
Коли створена стаття	Визначає дату створення статті
Контент	Визначає вміст статті
Файл тексту статті	Визначає вміст статті
Превью	Визначає зображення, прикріплене до статті

Продовження таблиці 2.1

Назва	**Визначення**
ВКЛ\ВИКЛ. коментарі	Визначає доступ до можливості коментування
Коли востаннє оновлена	Визначає, коли стаття була останній раз відредагована
Коментар	
Номер коментаря	Однозначно визначає номер кожного доданого коментаря
Ким написаний	Визначає користувачем, з яким номером написаний коментар
Коли написаний	Визначає дату додавання коментаря
Текст коментаря	Визначає вміст коментаря
Заявка на зміну ролі	
Логін	Однозначно визначає, хто залишив заявку
Контент	Визначає вміст заявки

2.3Оновлення структури бази даних

Об’єкти та типи сутностей

Грунтуючись на предметній області, для її всебічного опису було створено такі типи сутностей: користувачі, статті, коментарі. У поданій базі даних вони називаються так: USERS, ARTICLES, COMMENTS.

Усі сутності, окрім ARTICLES, залишилися незмінними. Зупинимося докладніше на описі цієї сутності, вказуючи на всі значущі моменти.

ARTICLES

Кожен екземпляр сутності articles має наступні характеристики:

Атрибут id (int (10)) – дає однозначний номер в базі даних, що належить конкретній статті. Його значення є унікальним в даній таблиці, тому цей атрибут є первинним ключем.

Атрибут name (varchar (255)) – описує ім’я автора певної статті.

Атрибут title (varchar (255)) – описує назву статті.

Атрибут content (text) – зберігає основний текст статті.

Атрибут preview (varchar (255)) – зберігає посилання-шлях до зображення, доданому до статті.

Атрибут public (tinyint (1)) – поле, в якому зберігається значення 0 або 1, що відповідно не дає запису бути опублікованим або дає і стаття відображається серед всіх.

Атрибут comments_enable (tinyint (1)) – поле, в якому зберігається значення 0 або 1, що відповідно не дає права коментування користувачами запис або дає таке право.

Атрибут created_at (timestamp) – поле, що визначає дату створення статті.

Атрибут updatetd_at (timestamp) – поле, що визначає дату зміни статті.

Атрибут textFile(varchar (255)) – зберігає посилання-шлях до текстового файлу, доданому до статті.

2.4 Розробка додаткового програмного забезпечення

Періодичне онлайн видання користується великим попитом, але функціонал сайту не є ідеальним. Для того щоб залучити більше користувачів до використання нашого продукту, було вирішено оптимізувати та автоматизувати деякий функціонал сайту.

2.4.1 Оптимізація роботи користувачів

Під оптимізацією роботи користувача онлайн видання мається на увазі оновлення або доробка функціоналу сайту для більш комфортної роботи з додатком.

Реорганізація сторінки кабінету користувача

Переглянувши базові сторінки сайту, було помічено недолік у кабінеті користувача. А саме, не зручне розташування елементів адміністрування. Отже, першим кроком є реорганізація сторінки кабінету користувача. Далі можна побачити сторінки у попередньому стані (рис. 2.1 – 2.3).

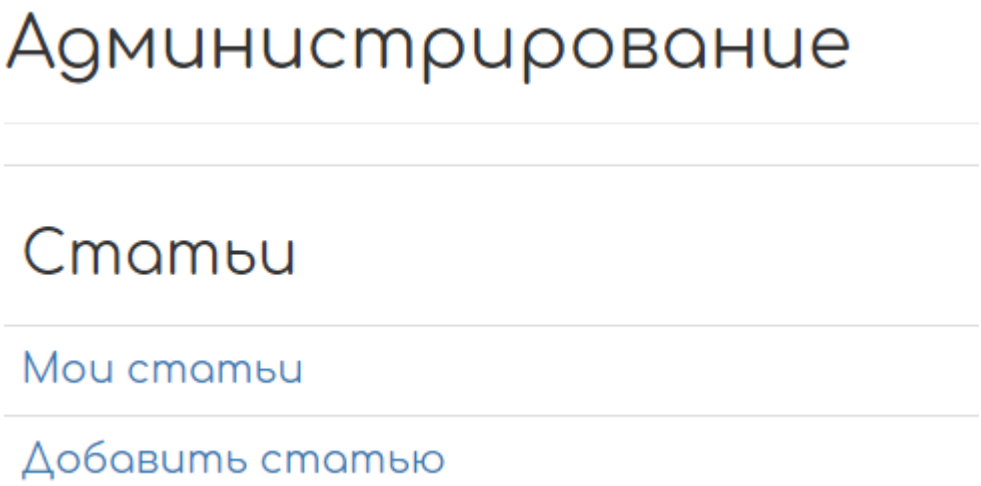


Рисунок 7.1 – Кабінет автора

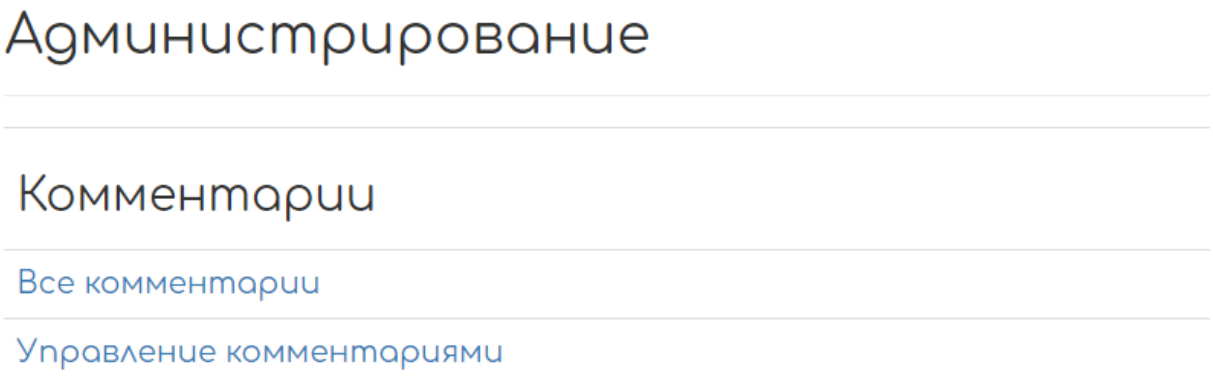


Рисунок 2.2 – Кабінет модератора

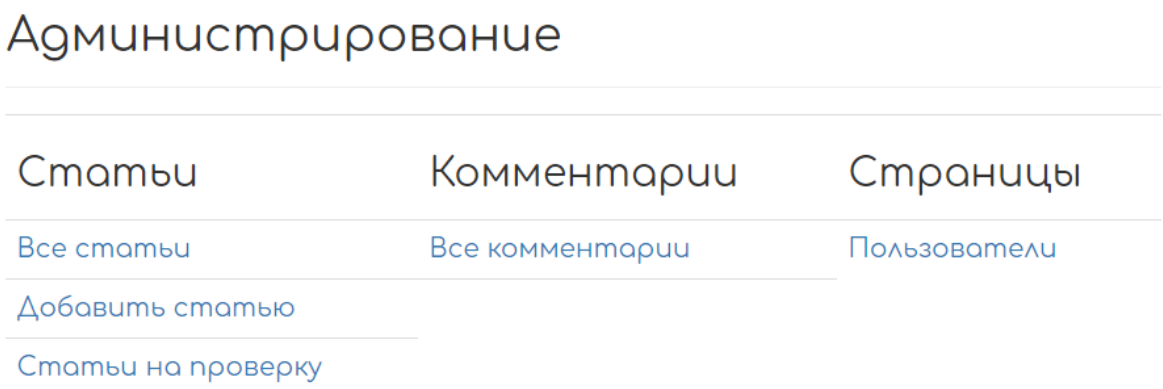


Рисунок 2.3 – Кабінет редактора

За рахунок оптимізації коду, сторінки завантажуються швидше. Усі зміни можна побачити у додатках (додаток А, Б). Вигляд оптимізованих сторінок можна побачити у розділі 3 (рис. 3.1 – 3.3).

Відображення статей автора

Раніше статті автора відображались наступним чином (рис. 2.4). Автор бачив статті усіх авторів, але мав змогу редагувати або видаляти тільки власні статті. Було створено окрему сторінку, яка відображує статті тільки певного автора. Результат оптимізації можна побачити у розділі 3 (рис. 3.4) та код сторінки у додатках (додаток В, Г).

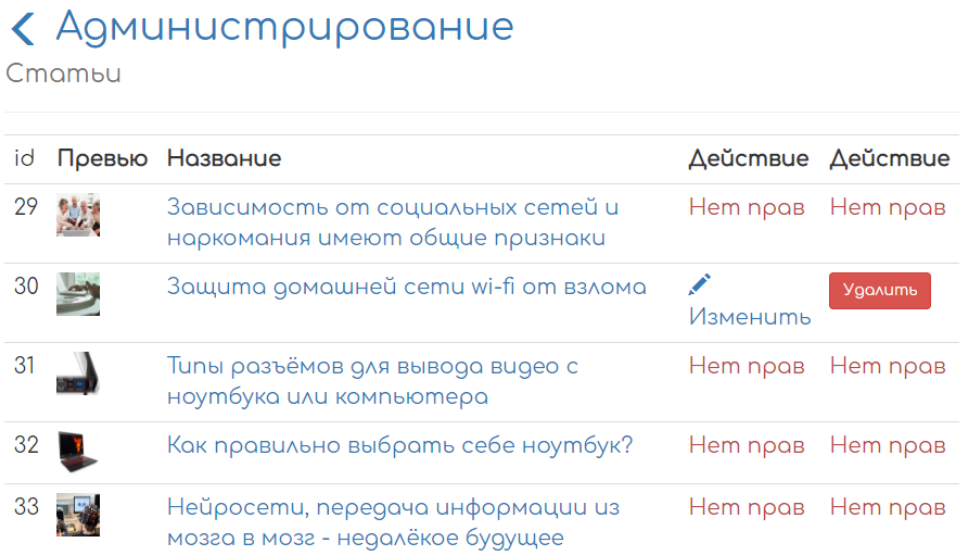


Рисунок 2.4 – Відображення статей автора до змін

Відео-уроки

Для оптимізації написання та редагування статей було створено розділ з відео-уроками. Відео носять навчальний та пізнавальний характер. Завдяки цьому автори та редактори додатку мають змогу поліпшити свої навички. Структуру та приклад сторінок з відео уроками можна побачити у розділі 3 (рис. 3.5 – 3.8). Код цих сторінок приведений у додатках (додаток Д, Е).

Статті на перевірку

Для зручності роботи редактора була змінена система подачі статті на редагування. Раніше редактору потрібно було перевіряти сторінку з усіма статтями на наявність нових статей, але після введених змін з’явилась сторінка зі статтями до редагування. Система має наступний алгоритм:

− автор пише статтю;

− за бажанням або потребою вказує, що статтю не потрібно одразу публікувати (рис. 2.5). Це означатиме, що стаття одразу потрапить на сторінку зі статтями до редагування);

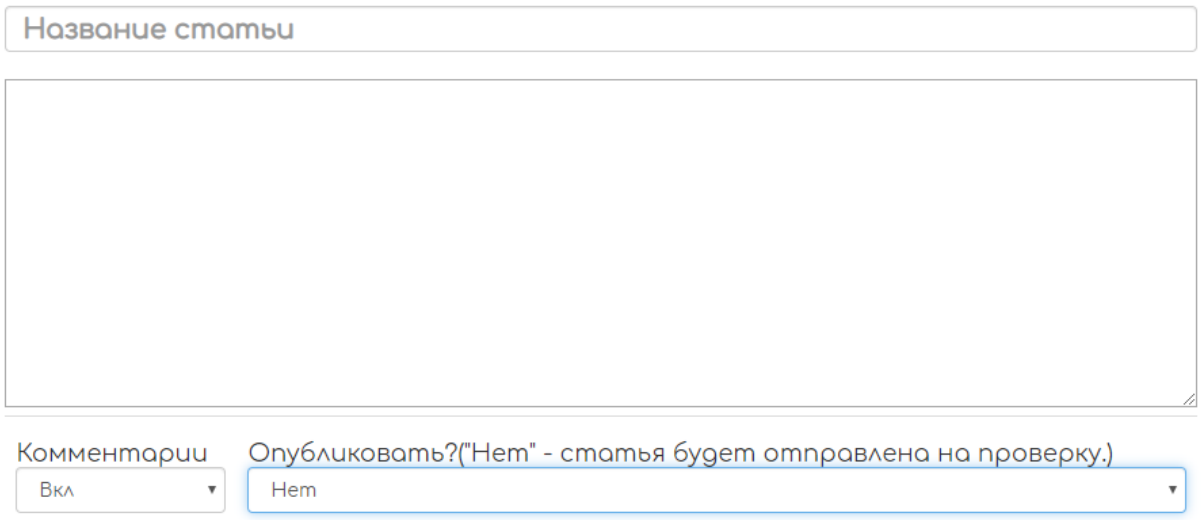


Рисунок 2.5 – Приклад відправлення статті на редагування

− редактор переходить на сторінку зі статтями до редагування, де відображуються тільки ті статті, які потрібно редагувати;

− редагує статтю;

− публікує статтю (стаття публікується від імені автора).

Побачити сторінку зі статтями до редагування можна побачити у розділі 3 (рис. 3.9, 3.10), а код сторінки у додатку (додаток Ж).

2.4.2 Автоматизація роботи користувачів

Додавання зображення до статті

Алгоритм завантаження зображення у базу даних був змінений, що дозволило оптимізувати зберігання файлів [13] у базі даних та автоматизувати роботу користувача. Наразі, маємо наступний алгоритм:

− перевіряємо чи було передано зображення, адже стаття може бути і без нього;

− визначається поточна дата, яка буде ім'ям каталогу для зображень;

− вказуємо кореневу папку для завантаження картинок;

− якщо папка з датою не існує, то створюємо її;

− переносимо файл у папку з оригінальним ім’ям;

− у змінну передаємо масив, який містить всі введені дані в формі;

− міняємо значення preview на посилання до каталогу;

− зберігаємо масив у базу;

− якщо зображення не передане, то зберігаємо запит як є.

Із кодом алгоритму можна ознайомитись у додатку (додаток И). У розділі 3 (рис. 3.11, 3.12) можна побачити, як додати зображення до статті.

Додавання файлу до статті

Деяким користувачам додатку не зручно писати статтю одразу у браузері. Вони віддають перевагу написанню статті у текстовому редакторі. Для автоматизації роботи цих користувачів була створена додаткова сторінка, на якій автор має змогу додати статтю із текстового файлу [15]. Базовий алгоритм роботи PHP із завантаженням файлів:

− користувач переглядає HTML-сторінку з формою, спеціально написаної для підтримки завантаження файлів;

− далі він надає файл, який він хоче завантажити і натискає кнопку відправити;

− браузер кодує файл і відправляє його як частину запиту POST;

− PHP отримує форму відправки, декодує файл і зберігає його в тимчасовому місці на сервері;

− написаний PHP-скрипт, відповідальний за обробку повідомлення форми, перевіряє файл і обробляє його якимось чином, часто переміщаючи його зі свого тимчасового місця розташування в постійне, де буде зберігатися файл.

Для додавання підтримки завантаження файлів необхідно створити HTML форму, яка буде представлена користувачеві, і скрипт PHP, який подбає про завантажений файл на сервері.

Код скрипту, форми з підтримкою завантаження файлів та сторінки з відображенням вмісту файлу можна побачити у додатках (додаток Л, М, Н).

Приклад роботи користувача з використанням завантаження статті із файлу наведено у розділі 3 (рис. 3.13 – 3.22).

3 ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Кабінет користувача

Далі можна побачити оновлений кабінет користувача (рис. 3.1 – 3.4).

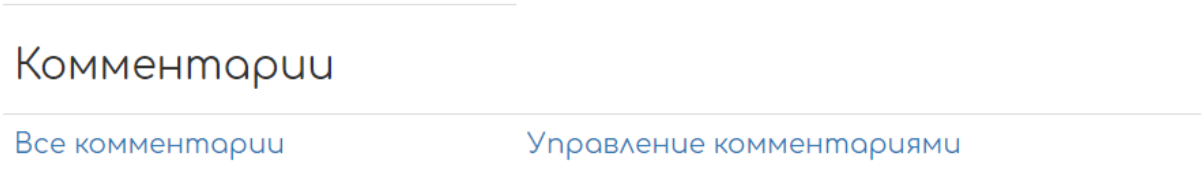


Рисунок 3.1 – Кабінет модератора

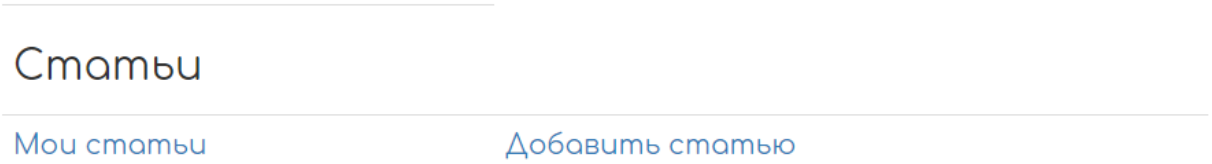


Рисунок 3.2 – Кабінет автора

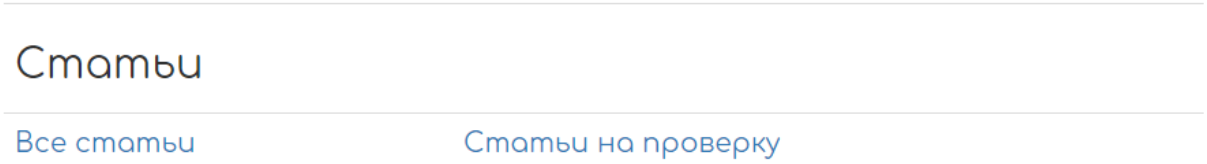


Рисунок 3.3 – Кабінет редактора

Кожен автор бачить тільки свої статті для видалення або редагування (рис. 3.4).

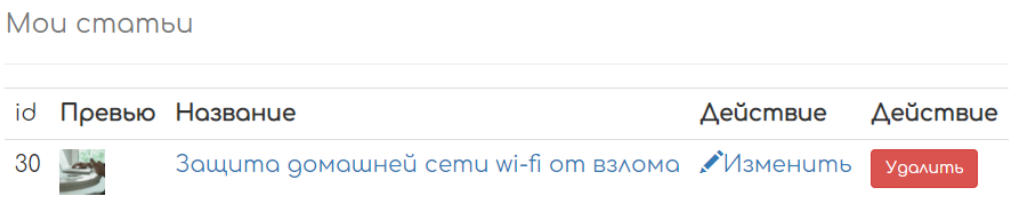


Рисунок 3.4 – Відображення статей певного автора

3.2 Відео-уроки

Структура сторінки з відео-уроками наступна (рис. 3.5):

- SEO оптимізація тексту;
- якісне написання статей;
- вичитування статей.

Приклад розташування відео на сторінці можна побачити нижче (рис. 3.6 – 3.8).

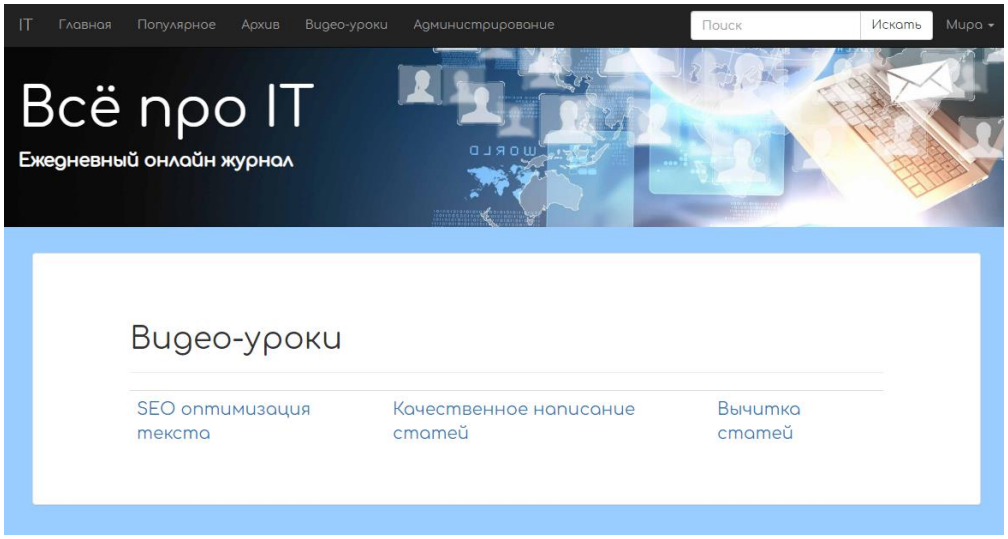


Рисунок 3.5 – Сторінка з відео-уроками

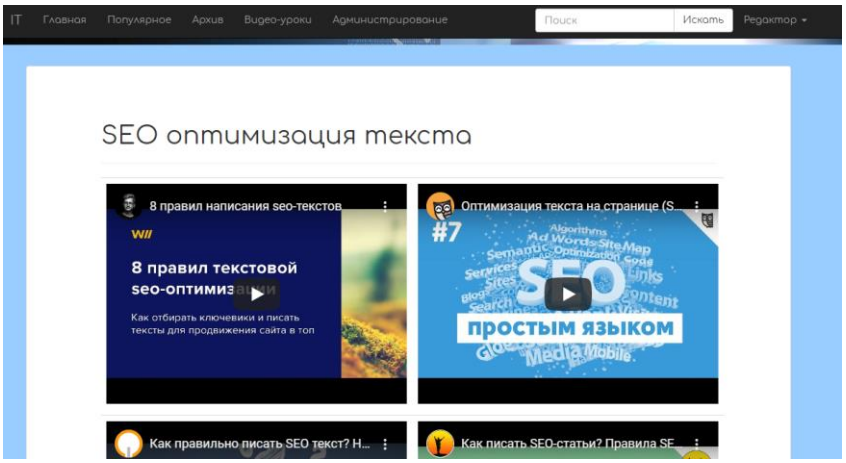


Рисунок 3.6 – відео-уроки з SEOоптимізації тексту

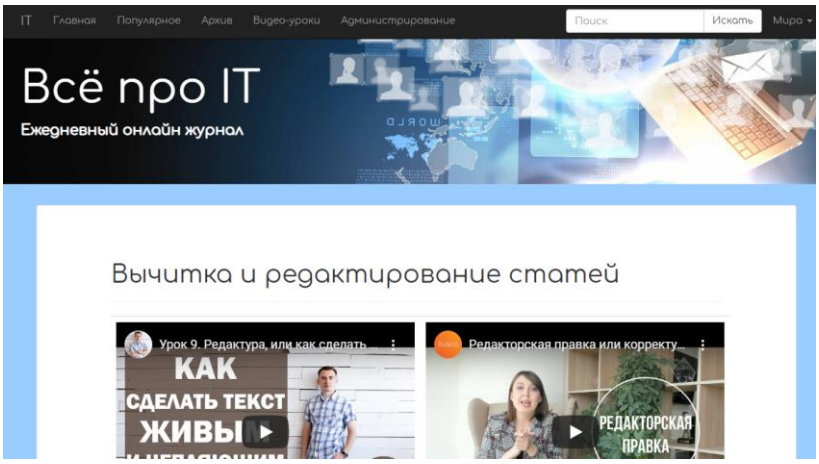


Рисунок 3.7 – відео-уроки з редагування статей

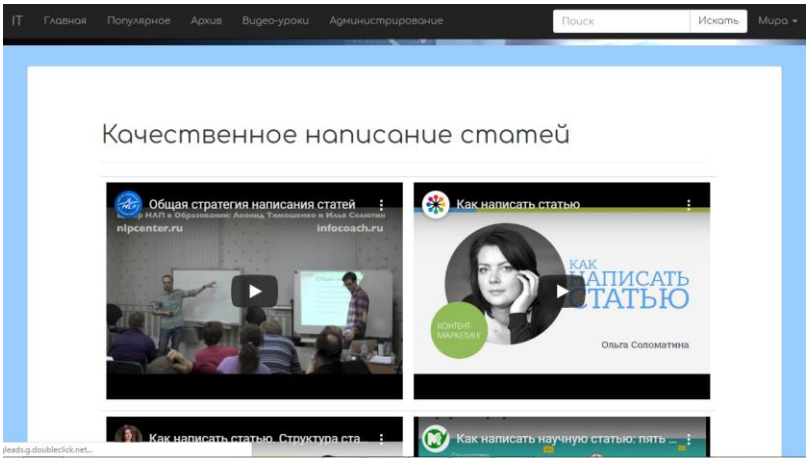


Рисунок 3.8 – відео-уроки з написання статей

3.3 Статті на перевірку

У кабінеті редактора додали нову сторінку, яка містить статті на перевірку (рис. 3.9, 3.10).

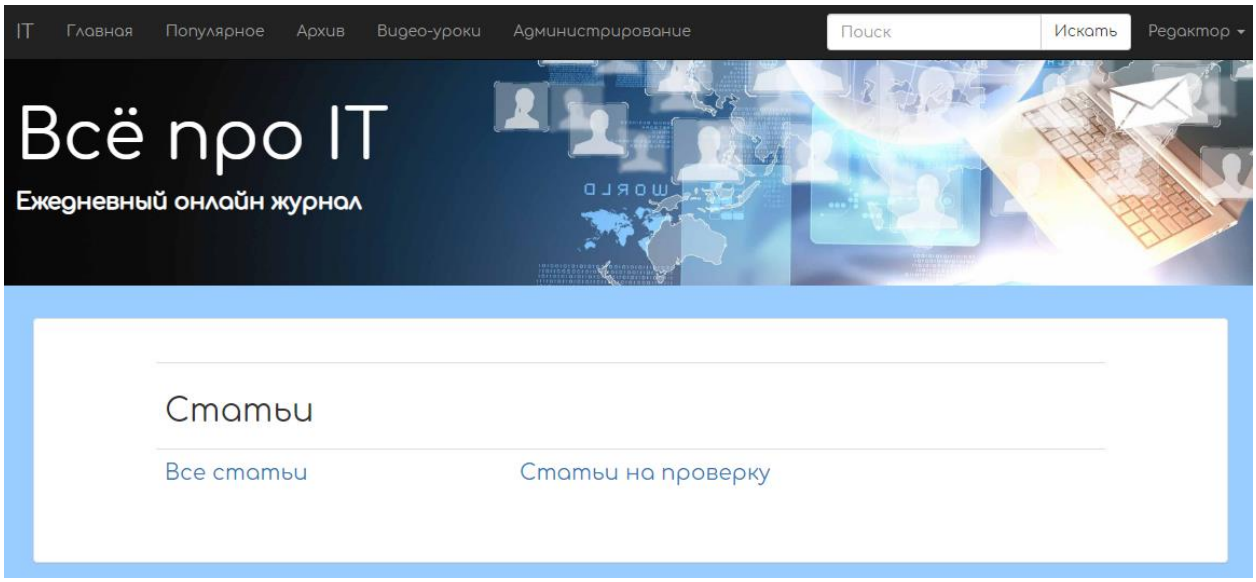


Рисунок 3.9 – Перехід до сторінки зі статтями до редагування

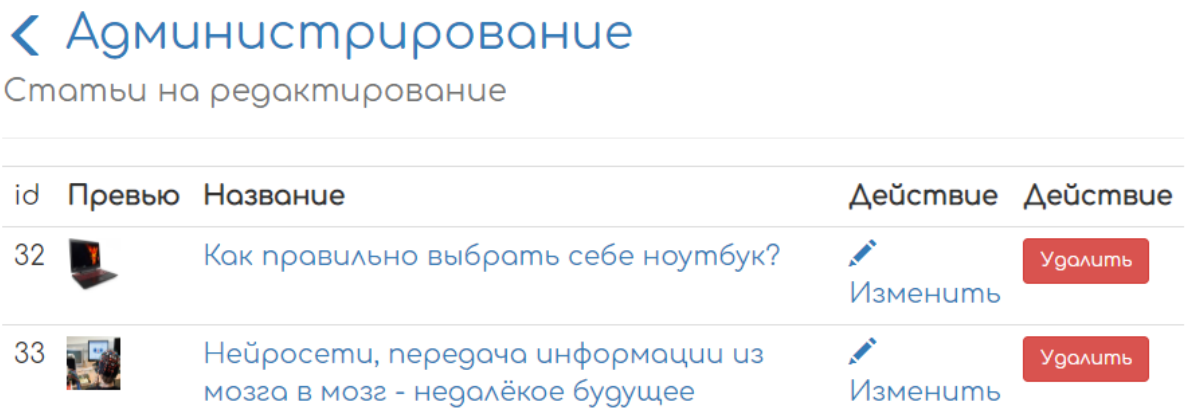


Рисунок 3.10 – Приклад відображення статей до редагування

3.4 Додавання зображення до статті

Кожен автор має змогу просто та швидко додати зображення до статті (рис. 3.11), яке потім буде відображатися у веб-додатку (рис. 3.12).

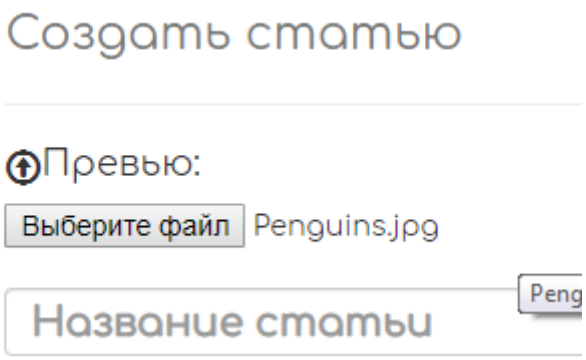


Рисунок 8.11 – Вибір зображення для завантаження

Рисунок 3.12 – Відображення зображення на сторінці додатку

3.5 Додавання файлу статті

Кожен автор має змогу додавати статтю із файлу. Не всі користувачі бажають так робити, тому було добавлено окрему сторінку, яка надає таку можливість (рис. 3.13). Приклад додавання статті із файлу можна побачити далі (рис. 3.14 – 3.22).

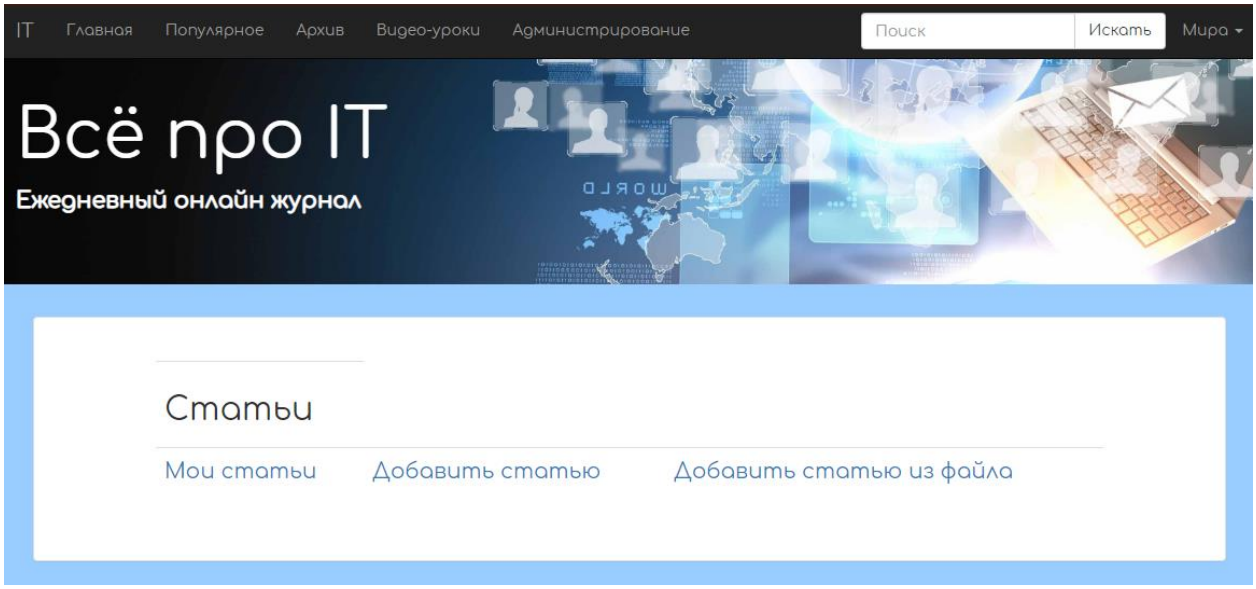


Рисунок 3.13 – Перехід до сторінки з додаванням статті із файлу

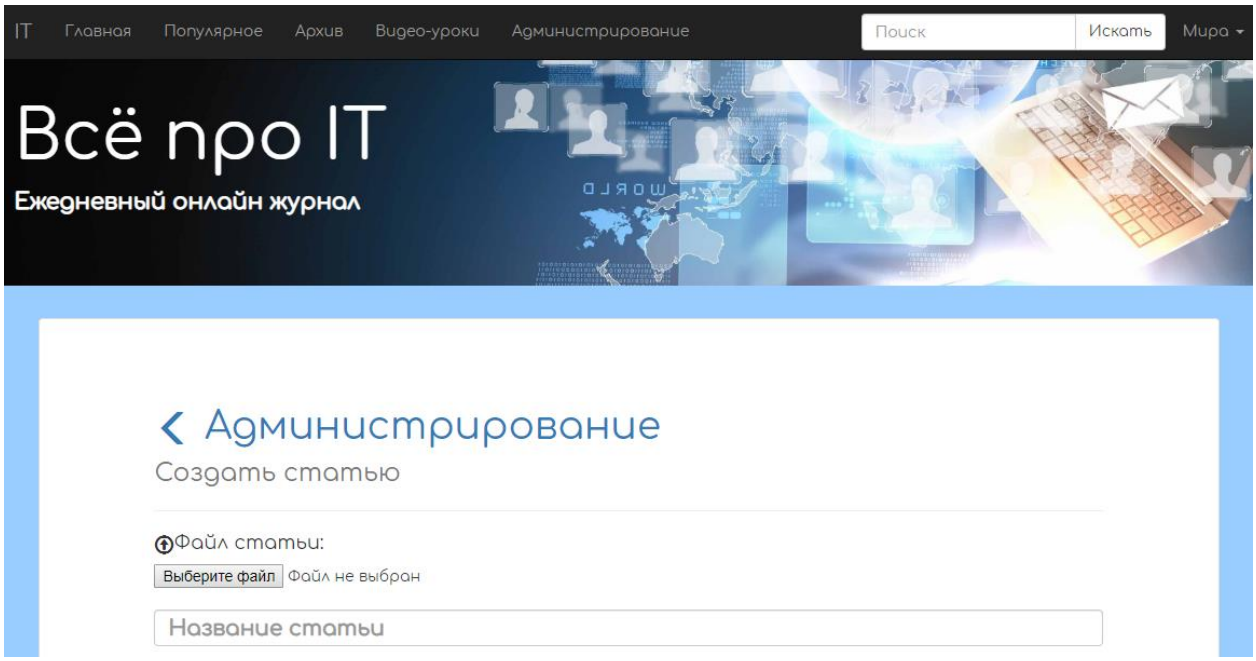


Рисунок 3.14 – Форма з підтримкою додавання статті із файлу

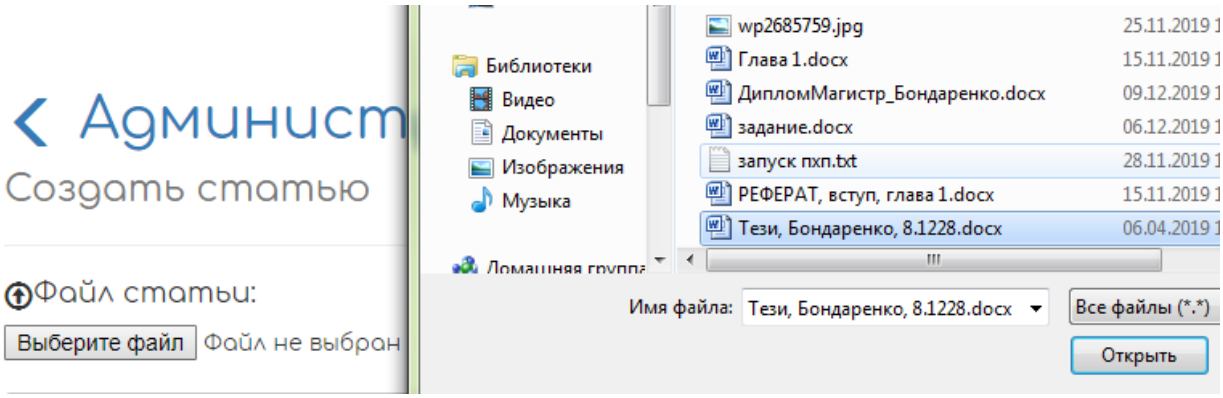


Рисунок 3.15 – Вибір файлу для завантаження

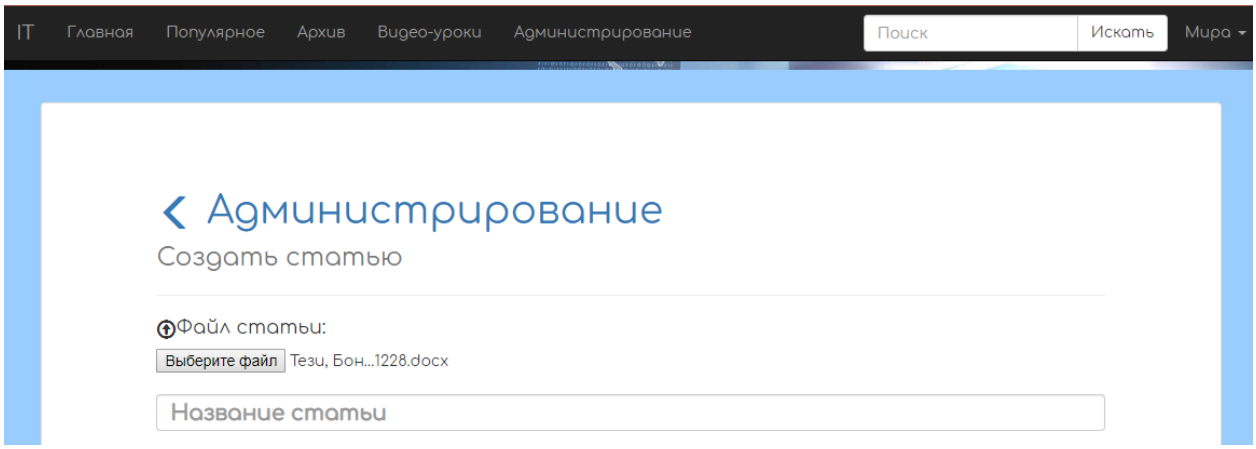


Рисунок 3.16 – Обраний файл готовий до завантаження

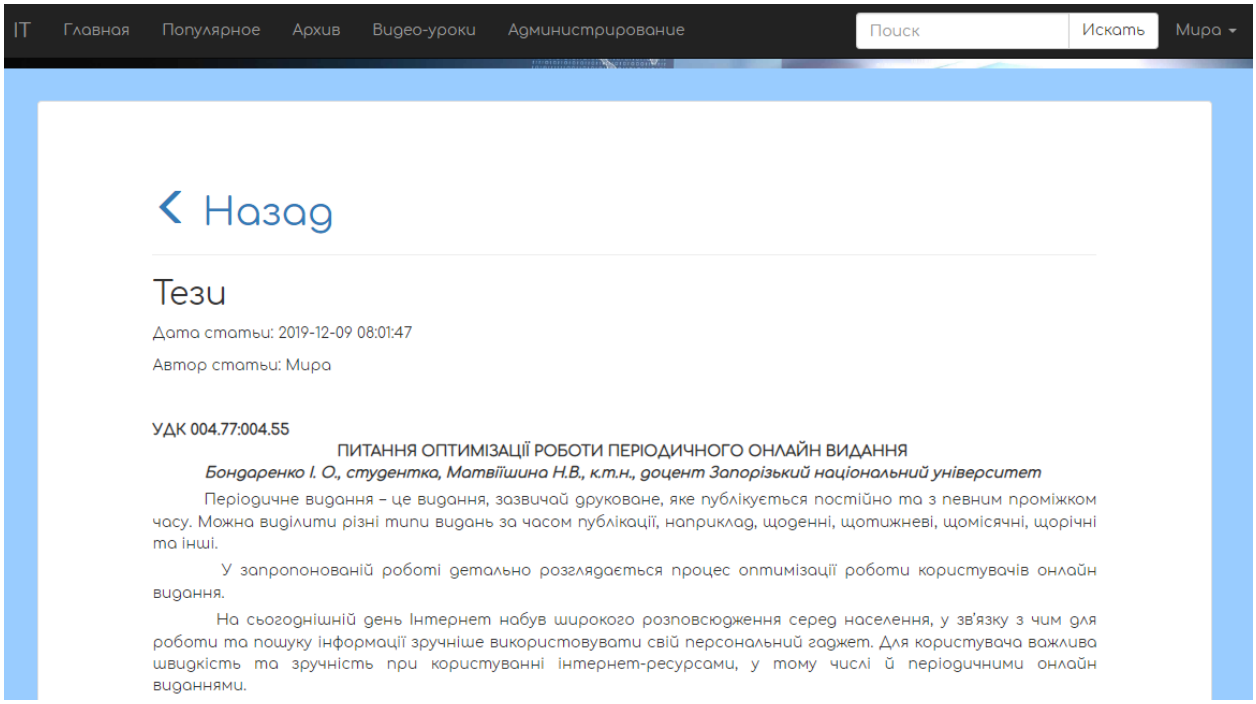
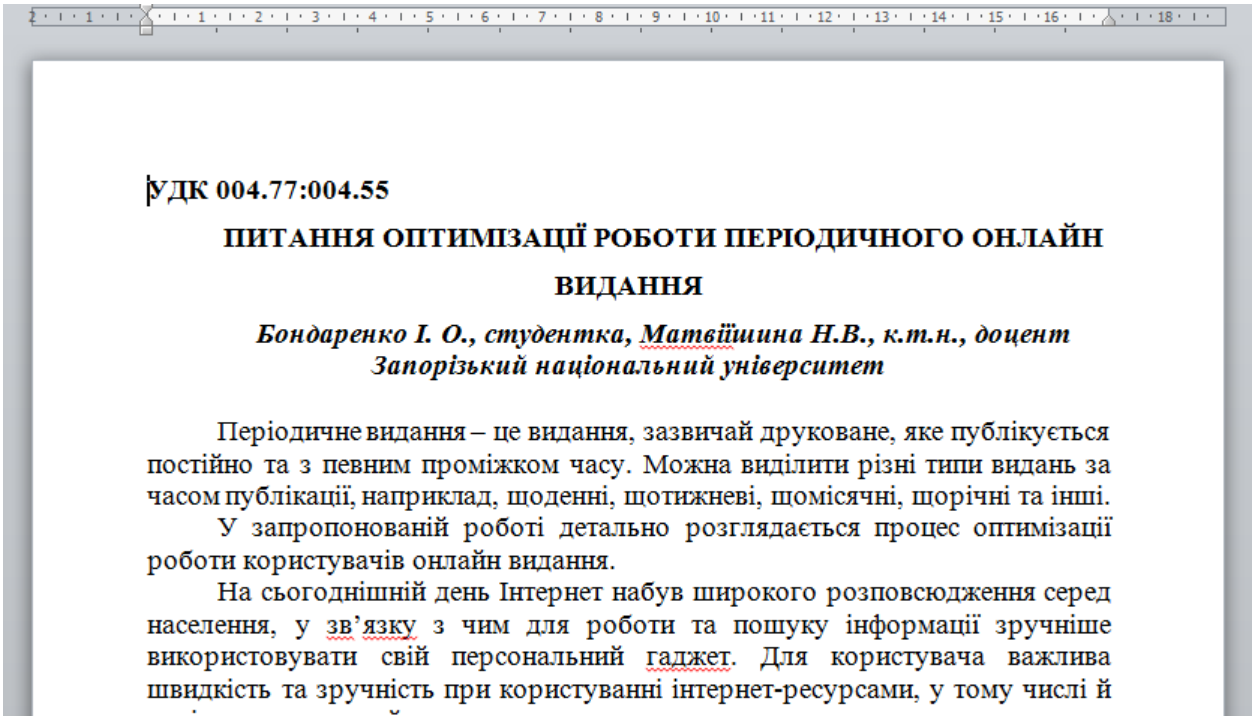


Рисунок 3.17 – Відображення змісту файлу на сторінці



УДК 004.77:004.55

ПИТАННЯ ОПТИМІЗАЦІЇ РОБОТИ ПЕРІОДИЧНОГО ОНЛАЙН ВИДАННЯ

Бондаренко І. О., студентка, Матвіїшина Н.В., к.т.н., доцент
Запорізький національний університет

Періодичне видання – це видання, зазвичай друковане, яке публікується постійно та з певним проміжком часу. Можна виділити різні типи видань за часом публікації, наприклад, щоденні, щотижневі, щомісячні, щорічні та інші.

У запропонованій роботі детально розглядається процес оптимізації роботи користувачів онлайн видання.

На сьогоднішній день Інтернет набув широкого розповсюдження серед населення, у зв'язку з чим для роботи та пошуку інформації зручніше використовувати свій персональний гаджет. Для користувача важлива швидкість та зручність при користуванні інтернет-ресурсами, у тому числі й

Рисунок 3.18 – Зміст завантаженого файлу

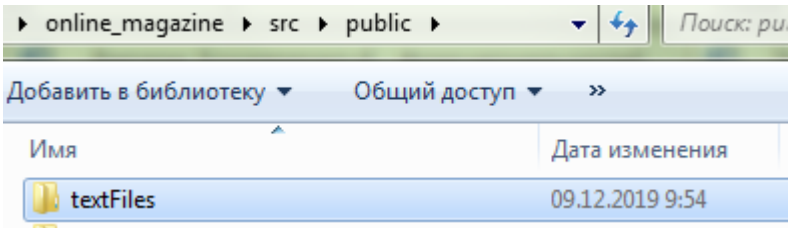


Рисунок 3.19 – Розташування текстових файлів на сервері

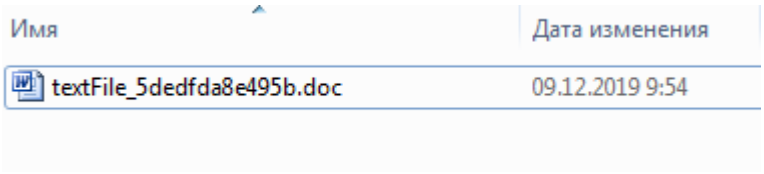


Рисунок 3.20 – Завантажений файл на сервері

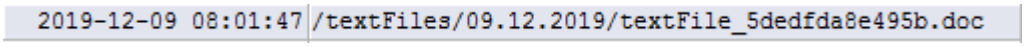


Рисунок 3.21 – Посилання на файл у базі даних

УДК 004.77:004.55

ПИТАННЯ ОПТИМІЗАЦІЇ РОБОТИ ПЕРІОДИЧНОГО ОНЛАЙН ВИДАННЯ

Бондаренко І. О., студентка, Матвіїшина Н.В., к.т.н., доцент
Запорізький національний університет

Періодичне видання – це видання, зазвичай друковане, яке публікується постійно та з певним проміжком часу. Можна виділити різні типи видань за часом публікації, наприклад, щоденні, щотижневі, щомісячні, щорічні та інші.

У запропонованій роботі детально розглядається процес оптимізації роботи користувачів онлайн видання.

На сьогоднішній день Інтернет набув широкого розповсюдження серед

Рисунок 3.22 – Зміст файлу на сервері

ВИСНОВКИ

У ході виконання кваліфікаційної роботи розроблено програмне забезпечення, яке автоматизує та оптимізує роботу користувача щоденного он-лайн видання (інтернет-журналу) «Всё про IT». Були виконані наступні задачі:

– аналіз літературних джерел з обраної теми;

– аналіз предметної області;

– створення опису предметної області, який містить інформацію про необхідні об’єкти та операції над ними;

– проектування додаткового програмного забезпечення;

– розробка програмного забезпечення;

– тестування оновленого функціоналуweb-додатку.

Виконання цієї роботи дозволило покращити вже існуючий функціонал та додати нові корисні функції (додавання статті із файлу, додавання зображення до статті) до веб-додатку, що дозволить залучити більшу аудиторію до користування щоденним он-лайн виданням, а також оновлений функціонал додатку сприятиме роботі вже існуючих користувачів. Функціонал був розроблений засобами мов PHP та JavaScript. Розроблений функціонал сайту має простий і зручний механізм роботи та відповідає вимогам, які були зазначені на першому етапі розробки проекту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Орлов Л. В. Web-сайт без секретов. Москва : Бук-пресс, 2006. 512 с.

2. Кузнецов М., Симдянов И. PHP. Практика создания Web-сайтов. Санкт-Петербург : БХВ-Петербург, 2008. 1264 с.

3. Фримен Э. Изучаем HTML, XHTML и CSS. Санкт-Петербург : Питер, 2010. 656 с.

4. Зандстра М. PHP : объекты, шаблоны и методики программирования. Москва : Вильямс, 2010. 560 с.

5. Зервас К. Web 2.0 : создание приложений на PHP. Москва : Вильямс, 2009. 544 с.

6. Лабберс П., Олберс Б., Салим Ф. HTML5 для профессионалов : мощные инструменты для разработки современных веб-приложений. Москва : Вильямс, 2011. 272 с.

7. Шафер С. HTML, XHTML и CSS. Библия пользователя, 5-е издание. Москва : Диалектика, 2010. 656 с.

8. Laravel по-русски. URL: https://laravel.ru/ (дата звернення 15.10.2019).

9. NicEdit. URL: http://www.nicedit.com/index.php (дата звернення 15.10.2019).

10. Мациевский Н., Степанищев Е., Кондратенко Г. Клиентская оптимизация в алгоритмах и примерах: Учебное пособие. Москва: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2010. 336 с.

11. Хольцнер С. Ajax Библия программиста. Москва: Диалектика, 2009. 553 с.

12. Херман Д. Сила JavaScript. 68 способов эффективного использования JS. Санкт-Петербург: Питер, 2013. 288 с.

13. Upload файлов, и все с этим связанное. PHP.SU. URL: http://www.php.su/articles/?cat=protocols&page=001 (дата звернення 25.10.2019).

14. Использование AJAX запросов в Laravel. Все о WEB программировании. URL: https://web-programming.com.ua/ispolzovanie-ajax-zaprosov-v-laravel/ (дата звернення 30.10.2019).

15. Laravel: работа с файлами – просмотр и удаление файлов. Все о WEB программировании. URL: https://web-programming.com.ua/laravel-rabota-s-fajlami-prosmotr-i-udalenie-fajlov/ (дата звернення 10.11.2019).

16. Загрузка файлов с помощью PHP на сервер. Falbar. URL: http://falbar.ru/article/zagruzka-fajlov-s-pomoshhyu-php-na-server (дата звернення 18.11.2019).

17. JavaScript. URL: http://crockford.com/javascript/ (дата звернення 28.11.2019).

ДОДАТОК А

Код сторінки кабінету користувача до змін

```
<table class="table" style="font-size: 20px">
<tr>
	@if (Auth::user()->role != 4 )
<td><h2>Статьи</h2></td>
	@endif
	@if (Auth::user()->role != 5 )
<td><h2>Комментарии</h2></td>
	@endif
	@if (Auth::user()->role != 4)
		@if (Auth::user()->role != 5)
<td><h2>Страницы</h2></td>
		  @endif
	@endif
</tr>
<tr>
	@if (Auth::user()->role != 4 )
	   @if (Auth::user()->role != 5 )
<td><a href="/adminzone/articles">Все статьи</a></td>
	@endif
	@endif
	   @if (Auth::user()->role == 5 )
<td><a href="/adminzone/authorarticles">Мои статьи</a></td>
	   @endif
	   @if (Auth::user()->role != 5 )
<td><a href="/adminzone/comments">Все комментарии</a></td>
```

```
            @endif
            @if (Auth::user()->role != 4 )
                @if (Auth::user()->role != 5)
<td><a href="/adminzone/users">Пользователи</a></td>
                    @endif
            @endif
</tr>
<tr>
        @if (Auth::user()->role != 4 )
<td><a href="/adminzone/articles/create">Добавить статью</a></td>
        @endif
<td>
 @if (Auth::user()->role != 2 )
 @if (Auth::user()->role != 5 )
<a                    href="{{action('CommentsController@show')}}">Управление
комментариями</a>
                    @endif
            @endif
</td>
 @if (Auth::user()->role == 1 )
<td><a href="/adminzone/rolesForm">Заявки на изменение роли</a></td>
            @endif
</tr>
<tr>
        @if (Auth::user()->role == 2 )
<td><a href="/adminzone/articlesforedit">Статьи на проверку</a></td>
        @endif
</tr>
</table>
```

ДОДАТОК Б

Код сторінки кабінету користувача після змін

```
@if (Auth::user()->role == 1 )
<div class="page-header">
<h1>Администрирование</h1>
</div>
<table class="table" style="font-size: 20px">
<tr>
<td><h2>Статьи</h2></td>
<td><h2>Комментарии</h2></td>
<td><h2>Страницы</h2></td>
</tr>
<tr>
<td><a href="/adminzone/articles">Все статьи</a></td>
<td><a href="/adminzone/comments">Все комментарии</a></td>
<td><a href="/adminzone/users">Пользователи</a></td>
</tr>
<tr><td><a href="/adminzone/articles/create">Добавить статью</a></td>
<td>
<a href="{{action('CommentsController@show')}}">Управление комментариями</a>
</td>
<td><a href="/adminzone/rolesForm">Заявки на изменение роли</a></td>
</tr>
<tr>
<td><a href="/adminzone/articlesforedit">Статьи на проверку</a></td>
</tr>
```

```
</table>
@endif
@if (Auth::user()->role == 2 )
<table class="table" style="font-size: 20px">
<tr>
<td colspan="2"><h2>Статьи</h2></td>
</tr>
<tr>
<td><a href="/adminzone/articles">Все статьи</a></td>
<td><a href="/adminzone/articlesforedit">Статьи на проверку</a></td></td>
</tr>
</table>
@endif
@if (Auth::user()->role == 4 )
<table class="table" style="font-size: 20px">
<tr>
<td><h2>Комментарии</h2></td>
</tr>
<tr>
<td><a href="/adminzone/comments">Все комментарии</a></td>
<td>
<a href="{{action('CommentsController@show')}}">Управление
комментариями</a>
</td>
</tr>
</table>
@endif
@if (Auth::user()->role == 5 )
<table class="table" style="font-size: 20px">
<tr>
```

```
<td><h2>Статьи</h2></td>
</tr>
<tr>
<td><a href="/adminzone/authorarticles">Мои статьи</a></td>
<td><a href="/adminzone/articles/create">Добавить статью</a></td>
</tr>
</table>
@endif
```

ДОДАТОК В

Відображення статей автора до змін

```
<table class="table" style="font-size: 20px">
<tr>
<td>id</td>
<td><b>Превью</b></td>
<td><b>Название</b></td>
<td><b>Действие</b></td>
<td><b>Действие</b></td>
</tr>
   @foreach ($articles as $article)
<tr>
<td>{{$article->id}}</td>
<td><img width=40 height=40 src="{{$article->preview}}"></td>
<td><a href="{{action('FrontController@show',['id'=>$article->id])}}">
                {{$article->title}}
</a></td>
<td>
            @if (Auth::user()->name == "$article->name")
<a href="{{action('ArticlesController@edit',['articles'=>$article->id])}}">
<span class="glyphicon glyphicon-pencil"></span>Изменить
</a>
            @else
<span style="color: #b94a48">Нет прав</span>
            @endif
</td>
<td>
```

```
            @if (Auth::user()->name == "$article->name")
<form                                                    method="POST"
action="{{action('ArticlesController@destroy',['articles'=>$article->id])}}">
<input type="hidden" name="_method" value="delete"/>
<input type="hidden" name="_token" value="{{csrf_token()}}"/>
<input type="submit" class="btn btn-danger" value="Удалить"/>
</form>
            @else
<span style="color: #b94a48">Нет прав</span>
            @endif
</td>
</tr>
   @endforeach
</table>
```

ДОДАТОК Г

Код сторінки з відображенням списку статей

```
<table class="table" style="font-size: 20px">
<tr>
<td>id</td>
<td><b>Превью</b></td>
<td><b>Название</b></td>
<td><b>Действие</b></td>
<td><b>Действие</b></td>
</tr>
   @foreach ($articles as $article)
      @if (Auth::user()->name == "$article->name")
<tr>
<td>{{$article->id}}</td>
<td><img width=40 height=40 src="{{$article->preview}}"></td>
<td><a href="{{action('FrontController@show',['id'=>$article->id])}}">
                {{$article->title}}
</a></td>
<td>
<a href="{{action('ArticlesController@edit',['articles'=>$article->id])}}">
<span class="glyphicon glyphicon-pencil"></span>Изменить
</a>
</td>
<td>
<form                                                    method="POST"
action="{{action('ArticlesController@destroy',['articles'=>$article->id])}}">
<input type="hidden" name="_method" value="delete"/>
```

```
<input type="hidden" name="_token" value="{{csrf_token()}}"/>
<input type="submit" class="btn btn-danger" value="Удалить"/>
</form>
</td>
</tr>
        @endif
    @endforeach
</table>
```

ДОДАТОК Д

Код сторінки з розділами відео-уроків

```
@extends('admin.main')
@section('content2')
<div class="page-header">
<h1>Видео-уроки</h1>
</div>
<table class="table" style="font-size: 20px">
<tr>
<td><a href="seo">SEO оптимизация текста</a></td>
<td><a href="writting">Качественное написание статей</a></td>
<td><a href="editting">Вычитка и редактирование статей</a></td>
</tr>
</table>
@endsection
```

ДОДАТОК Е

Код сторінки з відео-уроками

```
<div class="page-header">
<h1>SEO оптимизация текста</h1>
</div>
<table class="table" style="font-size: 20px">
<tr>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/t_Iz1Y8rESY" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/mQOpL8ij4nw" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
</tr>
<tr>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/R_5ZEuggzy4" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/peak7H-2V1g" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
</tr>
<tr>
```

```
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/4pDLJogoF_g" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/_CEl4Y0_vgg" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
</tr>
<tr>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/R6Hy4fgqoyY" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
<td><iframe width="430" height="315" src="https://www.youtube.com/embed/m-5qbDwmMrI" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></td>
</tr>
<tr><td></td><td></td></tr>
</table>
```

ДОДАТОК Ж

Код сторінки зі статтями до редагування

```
@extends('admin.main')
@section('content2')
<div class="page-header">
<h1>
<a href="/adminzone">
<span class="glyphicon glyphicon-menu-left" style="vertical-align: middle" aria-hidden="true"></span>
            Администрирование
</a><br>
<small>Статьи на редактирование</small></h1>
</div>
<table class="table" style="font-size: 20px">
<tr>
<td>id</td>
<td><b>Превью</b></td>
<td><b>Название</b></td>
<td><b>Действие</b></td>
<td><b>Действие</b></td>
</tr>
    @foreach ($articles as $article)
<tr>
<td>{{$article->id}}</td>
<td><img width=40 height=40 src="{{$article->preview}}"></td>
<td><a href="{{action('FrontController@showEditArt',['id'=>$article->id])}}">
                {{$article->title}}
```

```
</a></td>
<td><a href="{{action('ArticlesController@edit',['articles'=>$article->id])}}">
<span class="glyphicon glyphicon-pencil"></span>Изменить
</a></td>
<td>
<form method="POST"
action="{{action('ArticlesController@destroy',['articles'=>$article->id])}}">
<input type="hidden" name="_method" value="delete"/>
<input type="hidden" name="_token" value="{{csrf_token()}}"/>
<input type="submit" class="btn btn-danger" value="Удалить"/>
</form>
</td>
</tr>
      @endforeach
</table>
   @if(Session::has('message'))
      {{Session::get('message')}}
   @endif
@endsection
```

ДОДАТОКИ

Контролер завантаження зображень

```
public function store(Request $request)
{
    if ($request->hasFile('preview'))
    {
        $date = date('d.m.Y');
        $root = public_path() . "/images/";
        if (!file_exists($root . $date)) {
            mkdir($root . $date);
        }
        $f_name = uniqid('image_').'.jpg';
        $request->file('preview')->move($root . $date, $f_name);
        $all = $request->all();
        $all['preview'] = "/images/" . $date . "/" . $f_name;
        $article = Article::create($all);
    } else {
        $article = Article::create($request->all());
    }
    return redirect()->action(
        'FrontController@show', ['id' => $article->id]
    );
}
```

ДОДАТОК К

Відображення статті з зображенням

```
<?php
use App\User;
?>
@extends('admin.main')
@section('content2')
<div class="page-header">
<h1>
<a href="javascript:history.back(1)">
<span class="glyphicon glyphicon-menu-left" aria-hidden="true"></span>
            Назад
</a><br>
</h1>
</div>
<h2 class="blog-post-title">{{$article->title}}</h2>
<p class="blog-post-meta">Дата статьи: {{$article->updated_at}}</p>
<p class="blog-post-meta">Автор статьи: {{$article->name}} </p>
<p>
<img src="{{$article->preview}}" style="max-width: 80%;"><br>
<?=$article->content?>
   @if (Auth::guest())
   @elseif(in_array(Auth::user()->role,[User::ROLE_ADMINISTRATOR,
User::ROLE_EDITOR, User::ROLE_AUTHOR]))
<div style="float: right; display: block">
<form                                                    method="POST"
action="{{action('ArticlesController@destroy',['articles'=>$article->id])}}">
```

```
<input type="hidden" name="_method" value="delete"/>
<input type="hidden" name="_token" value="{{csrf_token()}}"/>
<h3><input type="submit" class="btn-lg btn-danger" value="Удалить"/>
           
; 
<a href="{{action('ArticlesController@edit',['articles'=>$article->id])}}">
                    Изменить <span class="glyphicon glyphicon-pencil"
style="vertical-align: middle"></span>
</a></h3>
</form>
</div>
      @endif
</p>
<br><br><br><br>
<hr>
<div class="comments" style="display:block;">
<h2>Комментарии</h2>
<table class="table" style="border: 0;">
          @foreach($comments as $comment)
<tr style="border-bottom: 1px solid #ddd">
<td>
<h4>
                {{$comment->author}} написал(а):<br>
           &
nbsp;   {{$comment->content}}<br>
<small> в {{$comment->created_at}}</small>
</h4>
</td>
</tr>
          @endforeach
```

```
</table>
</div>
<hr>
    @if(!Auth::guest() & $article->comments_enable==1)
        @include('site.comment')
        @endif
    @if(Auth::guest())
<h3>Чтобы добавлять комментарии <a href="/login">войдите</a> или <a href="/register">зарегистрируйтесь</a></h3>
    @endif
    @if($article->comments_enable==0)
<h3>Автор не захотел, чтобы эту статью комментировали.</h3>
    @endif
@endsection
```

ДОДАТОК Л

Контролер завантаження файлу статті

```
public function storeFromFile(Request $request)
{
    if ($request->hasFile('textFile'))
    {
        $date = date('d.m.Y');
        $root = public_path() . "/textFiles/";
        if (!file_exists($root . $date)) {
            mkdir($root . $date);
        }
        $f_name = uniqid('textFile_').'.doc';
        $request->file('textFile')->move($root . $date, $f_name
        $all = $request->all();
        $all['textFile'] = "/textFiles/" . $date . "/" . $f_name
        $article = Article::create($all);
    } else {
        $article = Article::create($request->all());
    }
    return redirect()->action(
        'FrontController@show', ['id' => $article->id]
    );
}
```

ДОДАТОК М

Форма з підтримкою завантаження файлів

```
<form method="POST" action="{{action('ArticlesController@storeFromFile')}}" enctype="multipart/form-data">
<h4><span class="glyphicon glyphicon-upload" style="vertical-align: middle"></span>Файл статьи:</h4>
<input type="file" name="textFile" required><br>
<input type="hidden" name="name" value="{{Auth::user()->name}}">
<input type="text" class="form-control" width="100%" name="title" style="font-size: 20px; font-weight: bold" placeholder="Название статьи" required>
<table class="table" style="border: 0">
<tr>
<td>
<h4>Комментарии<br>
<select name="comments_enable" class="form-control">
<option value="1">Вкл</option>
<option value="0">Выкл</option>
</select></h4>
</td>
<td>
<h4>Опубликовать?("Нет" - статья будет отправлена на проверку.)<br>
<select name="public" class="form-control">
<option value="0">Нет</option>
<option value="1">Да</option>
</select></h4>
</td>
</tr>
```

```
</table>
<h4>Мета</h4>
        description:
<input type="text" class="form-control" name="meta_description">
 keywords:
<input type="text" class="form-control" name="meta_keywords"><br>
<input type="hidden" name="_token" value="{{csrf_token()}}">
<br><input type="submit" class="btn-lg btn-primary" value="Сохранить">
</form>
```

ДОДАТОК Н

Відображення статті завантаженої з файлу

```
@if($file)
<h2 class="blog-post-title">{{$article->title}}</h2>
<p class="blog-post-meta">Дата статьи: {{$article->updated_at}}</p>
<p class="blog-post-meta">Автор статьи: {{$article->name}} </p>
<p>
<p>{{ $file }}</p>
@else
<h2 class="blog-post-title">{{$article->title}}</h2>
<p class="blog-post-meta">Дата статьи: {{$article->updated_at}}</p>
<p class="blog-post-meta">Автор статьи: {{$article->name}} </p>
<p>
<img src="{{$article->preview}}" style="max-width: 80%;"><br>
<?=$article->content?>
@endif
   @if (Auth::guest())
   @elseif(in_array(Auth::user()->role,[User::ROLE_ADMINISTRATOR,
User::ROLE_EDITOR, User::ROLE_AUTHOR]))
<div style="float: right; display: block">
<form                                                    method="POST"
action="{{action('ArticlesController@destroy',['articles'=>$article->id])}}">
<input type="hidden" name="_method" value="delete"/>
<input type="hidden" name="_token" value="{{csrf_token()}}"/>
<h3><input type="submit" class="btn-lg btn-danger" value="Удалить"/>
&amp;#160;&amp;#160;&amp;#160;&amp;#160;&amp;#160;&amp;#160;&amp;#160;&amp;#160;&amp;#160;&amp;#160
;&amp;#160;
```

```
<a href="{{action('ArticlesController@edit',['articles'=>$article->id])}}">
                    Изменить <span class="glyphicon glyphicon-pencil"
style="vertical-align: middle"></span>
</a></h3>
</form>
</div>
    @endif
```