Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка системи управління контентом голографічних 3D вітрин»

Студента 2 курсу, 7м групи
спеціальності
122 «Комп'ютерні науки»
спеціалізації
«Комп'ютерні науки»

_____
підпис студента

Кірова
Володимира
Віталійовича

Науковий керівник
доктор технічних наук, професор

_____
підпис керівника

Краскевич Валерій
Євгенович

Гарант освітньої програми
доктор фізико-математичних наук, професор

_____
підпис керівника

Пурський Олег
Іванович

Київ 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»
Спеціалізація «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«5» грудня 2019р.

Завдання
на випускний кваліфікаційний проект студенту

Кірову Володимиру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту
«Розробка системи управління контентом голографічних 3D вітрин»
Затверджена наказом ректора від *«02» грудня 2019 р. № 4110*

2. Строк здачі студентом закінченого проекту *05 листопада 2020 року*

3. Цільова установка та вихідні дані до проекту
Мета роботи: розробка системи управління контентом голографічної 3D вітрини з урахуванням мультиплатформенного користування.
Предмет дослідження: процес розробки системи управління контентом голографічної 3D вітрини.

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Краскевич В.Є.	5.12.2019 р.	5.12.2019 р..
2	Краскевич В.Є.	5.12.2019 р.	5.12.2019 р.
3	Краскевич В.Є.	5.12.2019 р.	5.12.2019 р.

6. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ЗМІСТ

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ КОНКУРЕНТОСПРОМОЖНОСТІ ПІДПРИЄМСТВА

1.1. Аналіз поняття CMS (система управління контентом)

1.2. Особливості функціонування голографічних 3D вітрин

1.3. Web-технології для CMS

1.4. Структура системи управління контентом

РОЗДІЛ 2. ОГЛЯД МЕТОДІВ СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ

2.1. Огляд та аналіз існуючих CMS

2.2. Огляд мов програмування web-систем

2.3. Вибір методики розробки CMS для голографічної 3D вітрини

РОЗДІЛ 3. СТВОРЕННЯ CMS ГОЛОГРІФІЧНОЇ 3D ВІТРИНИ

3.1. Розробка алгоритму створення CMS

3.2. Розробка web-системи для голографічної 3D вітрини

3.3. Підключення CMS

3.4. Правила роботи з CMS

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А

7. Календарний план виконання проекту

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*01.11.2019*	*01.11.2019*
2	*Розробка та затвердження завдання для випускного кваліфікаційного проекту*	*05.12.2019*	*05.12.2019*
3	*Вступ*	*01.06.2020*	*01.06.2020*
4	*РОЗДІЛ 1. Теоретичні аспекти системи управління контентом*	*25.06.2020*	*25.06.2020*
5	*РОЗДІЛ 2. Огляд методів створення системи управління контентом*	*02.09.2020*	*02.09.2020*
6	*Підготовка статті у збірник наукових статей магістрів*	*09.09.2020*	*09.09.2020*
7	*РОЗДІЛ 3. Створення CMS голографічної 3D вітрини*	*21.10.2020*	*21.10.2020*

8	*Висновки*	*02.11.2020*	*02.11.2020*
9	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*05.11.2020*	*05.11.2020*
10	*Попередній захист випускного кваліфікаційного проекту*	*20.11.2020*	*20.11.2020*
11	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*22.11.2020*	*22.11.2020*
12	*Представлення готового зшитого випускного кваліфікаційного проекту*	*25.11.2020*	*25.11.2020*
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «5» грудня 2019 р.

9. Керівник випускного кваліфікаційного проєкту
Краскевич В.Є.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Пурський О.І.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Кіров В.В.
(прізвище, ініціали, підпис)

12. Відгук керівника випускного кваліфікаційного проєкту

У роботі розглянемо web-технології для створення системи управління контентом (CMS), проведено аналіз існуючих систем управління контентом, розглянуті програмні засоби реалізації (мови, середовища розробки).
Створена CMS - голографічного 3d Вітрини. В результаті отримана CMS яка відтворює 3d моделі без використання сторонніх програм. Автор роботи є учасником кафедральної НДР НОМЕР державної реєстрації: 0119I100107.
Дипломний проект відповідає існуючим вимогам і може бути подана рекомендація щодо захисту. Професор д.т.н. Краскевич В.Е ..
Керівник випускного кваліфікаційного проекту
_____ Професор, доктор технічних наук Краскевич В.Е .
(підпис, дата)

13. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента Кіров В.В.
(прізвище, ініціали)
може бути допущений до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Пурський О.І.
(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2020 р.

АНОТАЦІЯ

Мета дослідження полягає в розробці системи управління контентом 3D вітрин за допомогою застосування сучасних web-технологій.

У процесі роботи досліджувалися різноманітні системи управління контентом.

Результатом роботи є проект створення системи управління контентом 3D вітрини.

Обсяг роботи: 49 сторінок, 25 ілюстрацій, 1 таблиця, 29 використаних джерел.

Ключові слова: 3D, CMS, голограма, система управління контентом.

ABSTRACT

The purpose of the study is to develop a content management system for 3D storefronts using modern web-technologies.

In the process of work, various content management systems were studied.

The result is a project to create a content management system for 3D showcases.

Volume of work: 49 pages, 25 illustrations, 1 table, 29 used sources.

Keywords: 3D, CMS, hologram, content management system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

CMS	—	Content management system.
HTML	—	HyperText Markup Language.
XHTML	—	Extensible HyperText Markup Language.
XML	—	Extensible Markup Language.
SGML	—	Standard Generalized Markup Language.
CSS	—	Cascading Style Sheets.

ВСТУП

Актуальність теми. Створення голографічного зображення це складний процес, що потребує знань в галузі інформатики, фізики, геометрії та дизайну. Для досягнення результатів використовується велика кількість специфічного обладнання та інформаційні технології. Саме тому голографічні проектори є, скоріш предметом розкоші, ніж продуктивною складовою реклами чи освіти. Використання 3D-технологій значно збільшить можливості голограми та зменшить собівартість пристроїв. Програмне забезпечення інформаційних технологій, що використовується в побудові та відтворенні голограм базується на web-дизайні, відеомонтажі та 3D-моделюванні.

Вдосконалення гаджетів, збільшення продуктивності та зменшення їх розмірів — це вектор розвитку сучасних інформаційних технологій. Новим напрямом мультимедійних презентацій стали голографічні призми та голографічні вітрини, а, зрештою, і голографія в цілому. Інформаційні технології, які використовуються в цих пристроях базуються на подачі зображень на спеціальні проекційні поверхні, які є прозорими або напівпрозорими, що дає змогу користувачам бачити зображення в просторі. Для досягнення ефекту об’ємного зображення використовують 3D-технологію та програмне забезпечення для відтворення 3D-зображення.

Для того, щоб 3D-зображення стало голографічним зображенням використовують спеціальні пристрої, такі як голографічні 3D-вітрини. За рахунок використання декількох екранів (як правило 3) та скляної поверхні з плівкою зворотної проекції, що розташовані під кутом нахилу 45 градусів, досягається ефект «голограми». За рахунок використання законів геометричної оптики та законів заломлення світла досягається ефект позиціонування зображення в просторі [1].

Метою роботи є практична реалізація системи управління контентом голографічної 3D вітрини з урахуванням мультиплатформенного користування.

Для досягнення поставленої мети необхідне вирішення наступних **завдань**:

- проаналізувати сутність задачі вибору системи управління контентом та проблеми її вирішення;
- дослідити методи розробки та вибору системи управляння контентом;
- розглянути елементи, що будуть відтворюватися за допомогою CMS (системи управляння контентом);
- розглянути постановку задачі розробки CMS;
- дослідити метод розробки та модифікації CMS;
- проаналізувати методи безперебійного мультиплатформенного доступу;
- визначити алгоритми розв’язання задачі створення CMS;
- розглянути вхідні та вихідні форми;
- визначити програмну реалізацію CMS.

Об'єктом дослідження виступає система управління контентом голографічної 3D вітрини.

Предметом дослідження є процес розробки системи управління контентом голографічної 3D вітрини.

РОЗДІЛ 1

ТЕОРЕТИЧНІ АСПЕКТИ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ

1.1. Аналіз поняття CMS (система управління контентом)

Система управління – програма, що надає інструменти для додавання, редагування, видалення інформації на сайті. Існують різноманітні системи управління сайтом, серед яких зустрічаються платні і безкоштовні, побудовані за різними технологіями. Кожен сайт має панель управління, яка є лише частиною всієї програми, але достатня для управління ним. Велика частина сучасних систем керування вмістом реалізується у вигляді візуального редактора – програми, яка створює HTML-код із спеціальної спрощеної розмітки, що дозволяє користувачеві простіше форматувати текст.

Основні завдання CMS такі:

1. Зібрати і об’єднати до єдиного цілого різнотипні джерела знань і інформації, які є доступними як усередині організації, так і за її межами.
2. Забезпечити взаємодію співробітників, робочих груп і проектів зі створеними ними базами знань, інформацією і даними так, щоб їх легко можна було знайти, витягнути і повторно використати в звичний для користувача спосіб.

Широкому впровадженню CMS сприяє достатньо багато причин. Найголовнішою є ускладнення функціональності сучасних сайтів, оскільки навіть пересічний власник сайту бажає, щоб на його сайті був і блог, і форум, і файловий архів. А потужні компанії потребують ще більших функціональних можливостей для своїх сайтів. Іншим важливим чинником стало спрощення самих CMS. Більшість хостингів пропонують установлення готових CMS безкоштовних чи комерційних версій. Прикладом є товариство Open Source, яке поширює безкоштовні системи управління сайтами. Вони будуть доречними для невеликих компаній, які не в змозі купити собі дорогий комерційний продукт [2].

Центральним елементом будь-якої CMS є сховища інформації. У сучасних системах управління контентом – це реляційна база даних. Слово “реляційна” вказує на те, що база складається з таблиць, між якими уставлені відносини. Якщо CMS необхідно зберегти певну інформацію, вона записує її в базу даних. Для кожної сутності в базі даних відведено окрему таблицю. Наприклад, таблиця, яка зберігає вміст Web-сторінок. У ній, окрім тексту сторінки, зберігається назва матеріалу, дата створення і відомості про автора. Поле “автор” посилається вже на таблицю користувачів, в якій містяться їх логіни, паролі і права. За допомогою встановлення таких модулів можна побудувати достатньо гнучку і надійну систему зберігання інформації.

Програмний рушій бази даних вибирається залежно від платформи. Якщо використовується платформа Windows, то це MS SQL, якщо UNIX платформа, то MYSQL. Після вибору бази даних варто визначитися, як краще запрограмувати роботу з нею в CMS. Кращим підходом є створення абстрактного прошарку роботи з базою даних. Реалізувати його можна як у вигляді спеціального класу, так і у вигляді набору функцій. В ідеалі основний код CMS має бути однаковим для будь-якої бази даних, змінюється лише код-прокладка для бази даних [3].

Інформацію, яку необхідно відобразити (наприклад, текст статті), CMS отримує з бази даних. Для відображення інформації у форматі HTML використовується механізм шаблонів. Шаблон є файлом з дизайном сторінки, що створено засобами спеціальної мови. Зазвичай, це певним чином розмічений код HTML, в якому вказано, де треба вставляти назву сторінки, де – основний текст, де – меню чи інші елементи, які беруться з бази даних. Найпростішим варіантом буде створення шаблону мовою PHP, але є і більш потужні рішення. Шаблонізатор має свою досить просту мову, з якою може впоратися верстальник, що не знає PHP. Шаблони перетворюються на файл PHP, а потім просто виконуються PHP-інтерпретатором. Для підвищення продуктивності можна скомпілювати шаблон у PHP, оскільки він буде рідко змінюватися. Таким чином, відбувається розподіл праці верстальника і програміста: один робить

шаблони, а інший пише код програми. Наступною частиною системи є система користувачів і їх ролей.

Роль користувача – це певний набір дій, які він може здійснювати. Ролі можна порівняти з групами користувачів в Windows. У сучасних CMS ролі користувача можна створювати і налаштовувати згідно з намірами розробника. Зазвичай, визначають кілька ролей користувачів: адміністратор, модератор, автор, користувач і відвідувач. Кожному користувачу можна надати певну роль, причому привласнення ролей відбувається або автоматично, або це робиться власноруч адміністратором. Перший варіант часто використовується на форумах, коли після досягнення певної кількості публікацій користувачу автоматично привласнюється новий статус [4].

1.2. Особливості функціонування голографічних 3D вітрин

Головною перевагою використання інформаційних технологій в голограмі є можливість імітації будь-якого об’єкту, незалежно від складності та розмірів. Таке рішення допомагає відтворити об’єкт та помістити його в потрібне середовище. Сфера застосування 3D-технологій не обмежена. Одним з рішень, де використовується технологія 3D-є голографічна 3D-вітрина «uScreener».

Чому саме 3D- та яка різниця у використанні 2D та 3D-зображень?

Принцип використання інформаційних технологій в голографії базується на взаємодії між собою мінімуму допоміжних пристроїв. Саме тому важливим елементом в голограмах є використання об’ємного 3D-зображення, що повністю замінює використання декількох екранів, дзеркал та скляних поверхонь [25, с.217–221].

Правила побудови голографічного зображення з використанням 2D-зображення:

1. Зображення з монітору подається на дзеркальну поверхню, яка розташована паралельно відносно екрану.

2. Інверсоване зображення з дзеркальної поверхні передається на скляну поверхню, яка розташована під кутом 45 градусів відносно дзеркальної поверхні. Таким чином утворюється об’ємне зображення, за рахунок подвійного відображення екрану на склі та екрану в дзеркалі.
3. Поверхня в голографічній вітрині має бути білого кольору. Саме цей колір дозволяє уникнути зайвих елементів, які знаходяться на екрані.
4. Використання плівки зворотної проекції дозволить уникнути небажаного засвітлення зображення та наддасть деталізації кольору.
5. Світло в корпусі голографічної вітрини допомагає освітити елемент та позиціонувати зображення відносно простору в середині вітрини.

Висновком, щодо використання 2D зображення є перевантажена деталями вітрина, яка повинна працювати в специфічних умовах, де кожна деталь не буде піддаватися впливу зовнішніх факторів та механічних пошкоджень, що важко досягається в умовах конференцій, виставок, презентацій. Інформаційні технології, що використовуються в проекціюванні 2D зображення відрізняються ускладненим системним кодом, порівняно з програмним забезпеченням, що використовується у проекціюванні 3D-зображення [1].

Правила побудови голографічного зображення з використанням 3D-зображення:

1. Зображення з монітору подається на скляну поверхню. На скляну поверхню нанесено плівку зворотної проекції.
2. Зображення зі скляної поверхні передається на задню стінку проектора, що в свою чергу допомагає уникнути будь-яких темних та яскравих променів. Таким чином на поверхню потрапляють лише кольорові промені.
3. Внутрішня частина виконана в чорних кольорах. Це допомагає надати внутрішнього об’єму.
4. Вертикальне освітлення в середині вітрини допомагає візуально зупинити відображення зі скла та створити ефект об’єму.

5. Використання 3D-зображення дозволяє уникнути використання додаткових дзеркал, та завдяки темному фону створити об’єм в середині вітрини.

1.3. Web-технології для CMS

Важливі критерії при виборі технологій:

1. Розмір і тип проекту.
2. Складність проекту.
3. Швидкість розробки.
4. Вартість фахівців.
5. Доступність фахівців.
6. Доступні інструменти розробки.
7. Наявність готових рішень.
8. Гнучкість рішення.
9. Наявність широкого спільноти.
10. Відмовостійкість рішення.
11. Тренд його розвитку.
12. Наявність детальної документації.
13. Вартість підтримки.
14. Вимоги до навантажень.
15. Вимоги до безпеки.
16. Кросплатформність.
17. Можливості інтеграції з іншими рішеннями.

Обираючи технологію за такими критеріями, можливо домогтися об'єктивного вибору і тим самим заощадити собі час і гроші [6].

Які бувають проекти

Часто тип проекту говорить сам за себе, і можна відразу сказати, що підійде: або вже готове рішення, або хоча б в який бік потрібно рухатися.

За складністю проекти поділяються:

1. Прості (візитки, Лендіги, прості інтернет-магазини, прості програми) - такі рішення зазвичай робляться на тематичних коробкових рішеннях, CMS або шаблонах.

2. Середні (складні інтернет-магазини і маркетплейси, портали національного масштабу, різноманітні сервіси, просунуті додатки) - такі рішення зазвичай робляться на фреймворках.

3. Складні (величезні портали, соціальні мережі, інноваційні та нетипові рішення) - ядро таких проектів зазвичай розробляється на чистому (нативном) мовою програмування [7].

За тематикою: інтернет-магазини, дошки оголошень, соціальні мережі і т.д. Для більшості популярних тематичних рішень вже давно є коробкові продукти.

У технологіях виділяють 3 рівня абстракції:

1. Чиста мову - це матеріал, з якого можна зробити все, що завгодно. Обмежують тільки можливості мови. На чистій мові зроблені всі найбільші сайти світу з відвідуваністю в сотні мільйонів і мільярдів користувачів, такі як: Instagram, YouTube, Pinterest, Tumblr, Dropbox, Twitter, Facebook, Amazon, Digg, LinkedIn та інші. Більш того, найбільші проекти в світі навіть створюють нові технології для себе, так як вже існуючі їх не влаштовують.
2. Фреймворк - це певне середовище розробки для програміста з готовими правилами і інструментами. Фреймворк, з одного боку, допомагає і прискорює розробку, а з іншого, накладає певні обмеження. На фреймворках робляться проекти середньої складності з відвідуваністю в мільйони.
3. CMS - це вже готове рішення, конструктор, в якому по частинах збирають потрібний проект. Його скоріше не програмують, а налаштовують. Обмежень тут величезна кількість, вийти за межі коробки складно і неефективно. На CMS робляться прості сайти з відвідуваністю до мільйона користувачів в місяць. Найчастіше один рівень абстракції базується на іншому. Тобто на чистому мовою роблять фреймворкі, а на

фреймворках роблять CMS. Для кожної популярної мови є багато різних фреймворков і CMS [8].

Сьогодні є величезна кількість різних мов програмування, на яких роблять сайти. І, більш того, на всіх популярних мовах є приклади величезних сайтів. Якщо 10 років тому, кажучи про технології великих сайтів, все говорили переважно про Java, то сьогодні це може бути майже будь-яку мову, і стверджувати, що сайти робляться на якомусь конкретному мовою, - стереотип. Це пов'язано з розвитком самих мов, за останнє десятиліття багато сильно просунулися в розвитку і отримали широкі можливості. Звичайно, кожна мова чимось відрізняється, і вибираючи її знову ж повинні керуватися об'єктивними критеріями з оглядкою на завдання проекту.

На чистій мовою, без використання фреймворков і коробкових рішень, пишуться величезні проекти з підвищеними вимогами по гнучкості, навантажень і безпеки. Для таких величезних проектів часто бюджет не грає такого значення, як ефективність. Чим більше проект, тим більше буде вимог по гнучкості і навантажень, а значить, простіше писати все з нуля, виділяючи на це кращих фахівців, ніж якщо брати якісь готові рішення, які незрозуміло ким писалися і незрозуміло які проблеми в них приховані. Наприклад, коли мова про невеликому проекті з відвідуваністю в 10 тис. Чоловік в день, то буде дешевше зробити його на CMS, яка буде споживати в 3 рази більше ресурсів сервера, поставити додатковий сервер за 50 $ / міс., і він буде працювати. Коли ж говорити про сайти з відвідуваністю в 100 млн. користувачів в день, вартість додавання серверів буде просто космічної, тому простіше і дешевше вкласти гроші в розробку рішення на чистій мові, яке буде оптимальним саме для конкретного проекту [9].

Чим більше проект, тим більше стек технологій, який в ньому використовується. У величезних порталах може використовуватися відразу кілька мов програмування. Знову ж таки, ми приходимо до об'єктивними критеріями вибору технологій. Часто одна мова може добре вирішувати одне завдання, а інший - іншу. Такі проекти можуть бути настільки величезними, що

їх частини можуть працювати на різних серверах, з різними доменами (піддоменами) і різними технологіями. Не слід боятися вінегрету технологій у великому проекті, хоча і допускати його потрібно тільки, коли це дійсно необхідно, а також пам'ятати, що далеко не всі технології сумісні. Найяскравіший приклад використання різних технологій - Google. Він на стільки великий, щоб різні його частини написані на C / C ++, Java, Python, JS та іншими мовами. Більш того, Google активно створює нові технології, як, наприклад, популярний нині AngularJS.

1.4. Структура системи управління контентом

Генерація сторінок за запитом. Системи такого типу працюють на основі зв'язки (рис. 1).

Контент-менеджер

↓

Модуль редагування

↓

База даних

↓

Модуль подання (ядро)

↓

Відвідувач

Рис. 1.1. Генерація сторінок за запитом

Інформація в базі даних змінюється за допомогою модуля редагування. Модуль подання генерує сторінку із змістом при запиті на нього, на основі

інформації з бази даних. Сторінки наново створюються сервером (модулем подання) при кожному запиті. На основі URL у модулі запиту відбувається визначення запитуваної порції контенту/сторінки (наприклад через параметри методу GET) [10].

Розповсюджені архітектурні і технічні помилки при розробці CMS

1. Розмежування на рівні сутностей БД елементів контенту і навігації (меню). Меню не є незалежною сутністю і фактично являє собою ієрархічний перелік посилань на елементи контенту, тобто вся інформація про меню міститься в об'єктах контенту. Зберігати ієрархію сторінок в таблиці веб-сторінок, за допомогою «рекурсивного зовнішнього ключа на цю ж таблицю» (дерево сторінок).
2. Використання в URL сторінки автоінкрементного ідентифікатору сторінки. Неефективно з точки зору usability. У якості унікального поля використовувати текстовий ідентифікатор сторінки (наприклад: about, contacts, pronas, tovary).
3. Залежність URL сторінки від місця сторінки в структурі сайту. Хоча ця властивість зазначається як позитивна з точки зору usability (видно по URL, де ця сторінка в структурі), це має негативне значення для потреби гнучкості розвитку сайту і SEO (оптимізації під пошукові системи) – у разі зміни URL сторінка втратить придбані раніше позиції в пошукових системах. Якщо ж URL сторінки не залежить від місця сторінки в ієрархії сайту, її можна вільно пересувати, розвиваючи ресурс, не втративши пошукових позицій. У якості *унікального* ідентифікатора стоірнки використовувати текстовий ідентифікатор (наприклад: about, contacts, pronas, tovary). У свою чергу це позитивно відобразиться на usability.
4. Використання у якості URL сторінки транслітерованого заголовка сторінки – негативно позначається для SEO: при зміні заголовка змінюється URL.

5. Багатомовність реалізується за допомогою додаткових сутностей (таблиць) в БД – ускладнює як роботу програміста, так і роботу контент-менеджера. Зберігати іншомовні версії в тій же таблиці сторінок у відповідних полях, що дублюють основні текстові поля для контенту [11].

РОЗДІЛ 2

ОГЛЯД МЕТОДІВ СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ КОНТЕНТОМ

2.1. Огляд та аналіз існуючих CMS

Існує величезна кількість різних CMS, на будь-який смак. За допомогою сучасних систем управління сайтом, можливо реалізувати будь-який проект, що потребує автоматизації. Різниця полягає тільки в ціні, є платні і безкоштовні системи. Замовник визначає сам, яку CMS використовувати. На користь безкоштовних систем варто відзначити наступне - професійна веб-студія, при розробці сайту, здатна реалізувати можливості, надані платними системами. Недолік створення сайту – це "вага", CMS розроблені на всі випадки життя, відповідно вони складаються з величезної кількості файлів [22].

Доступних CMS існує досить багато. Частина з них вже застаріла як морально, так і в плані технологій, а деякі давно не підтримуються навіть самими розробниками.

Опис най популярніших, серед розробників, CMS:

- 1С-Бітрікс

Ця CMS займає перше місце за популярністю серед платних движків інтернет-магазину, відрізняється поширеністю і популярністю, насамперед, в корпоративному секторі. Її відносять до систем промислового рівня. Для відповідності вимогам власника сайту і зручного управління інтернет-магазином, необхідно добре налаштувати сам движок. Про це говорить і програма сертифікації фахівців від розробників. Також дана платформа одна з

найбільш вимогливих до ресурсів сервера, і для стабільної роботи використання потужного хостингу є необхідністю.

Особливості 1С-Бітрікс:

- вартість ліцензії на редакції для інтернет-магазину: 9 845 гривень і 17 800 гривень;
- велика кількість опцій для користувальницької і адміністративної частини;
- маркетплейс з великою кількістю модулів і платних шаблонів;
- безшовна інтеграція з 1С для автоматизації роботи;
- велика кількість розробників, але ціни на роботу вищі за середньо ринкову;
- користувачі відзначають певну переобтяженість інтерфейсу управління і вимогливість системи до ресурсів сервера.

Фактично, вартість розробки інтернет-магазину на "Бітрікс" починається від 20 тисяч гривень, а середня ціна на рівні 70 тисяч гривень (рис 2.1) [23].

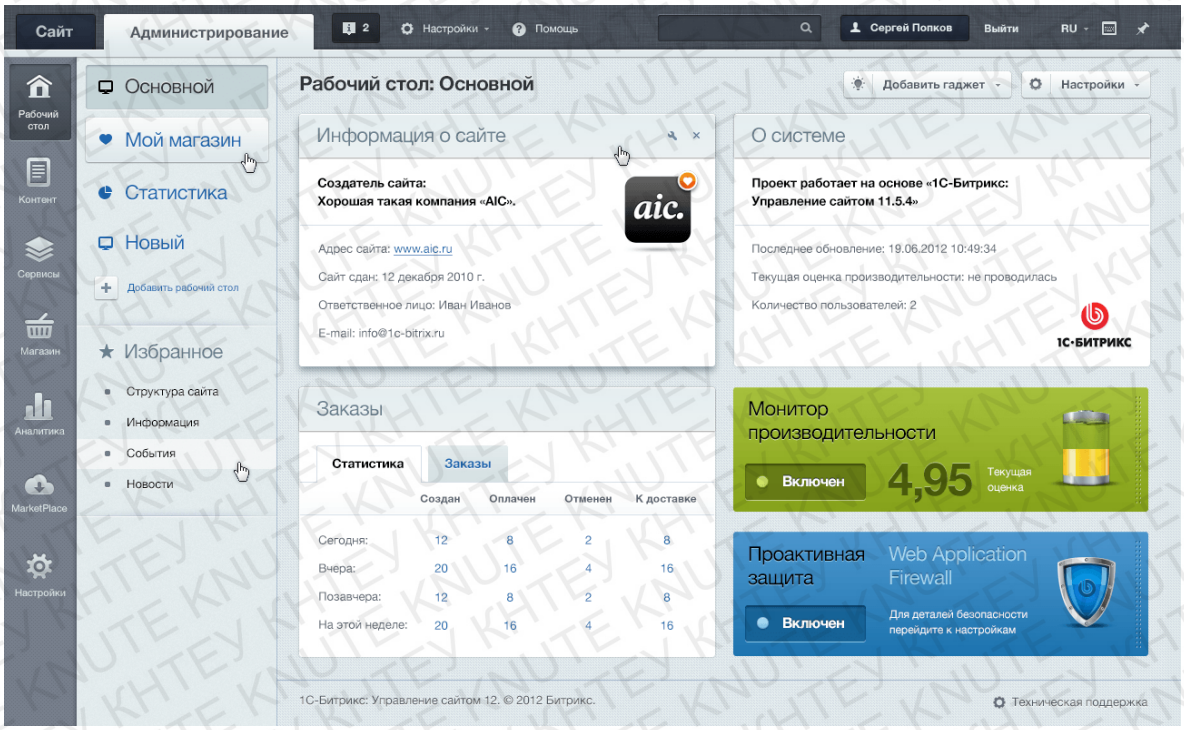


Рис. 2.1. Інтерфейс 1С-Бітрікс

- UMI.CMS.

Друге місце за популярністю серед розробників інтернет-магазинів займає UMI.CMS. Вартість редакцій з функціоналом електронної комерції

нижче, в порівнянні з «Бітрікс», - 8 тисяч гривень та 13 тисяч гривень. Інтерфейс системи простіше для освоєння новачком, і вимогливість до ресурсів менше. Десятки доступних модулів, але відсутнє маркетплейс доповнення. Нестандартні додаткові доповнення доведеться програмувати окремо.

Особливості UMI.CMS:

- нескладна в освоєнні адміністративна панель, додавання товарів і управління каталогом реалізовано досить зручно;
- велика техпідтримка, структурована документація як в текстовому, так і у відео форматі;
- в безкоштовному доступі шаблонів дизайну практично не знайти, платних теж мало;
- при великій кількості сторінок в структурі керувати сайтом стає незручно;
- відсутність маркетплейс доповнень, що накладає обмеження при розвитку проекту (рис 2.2) [24].

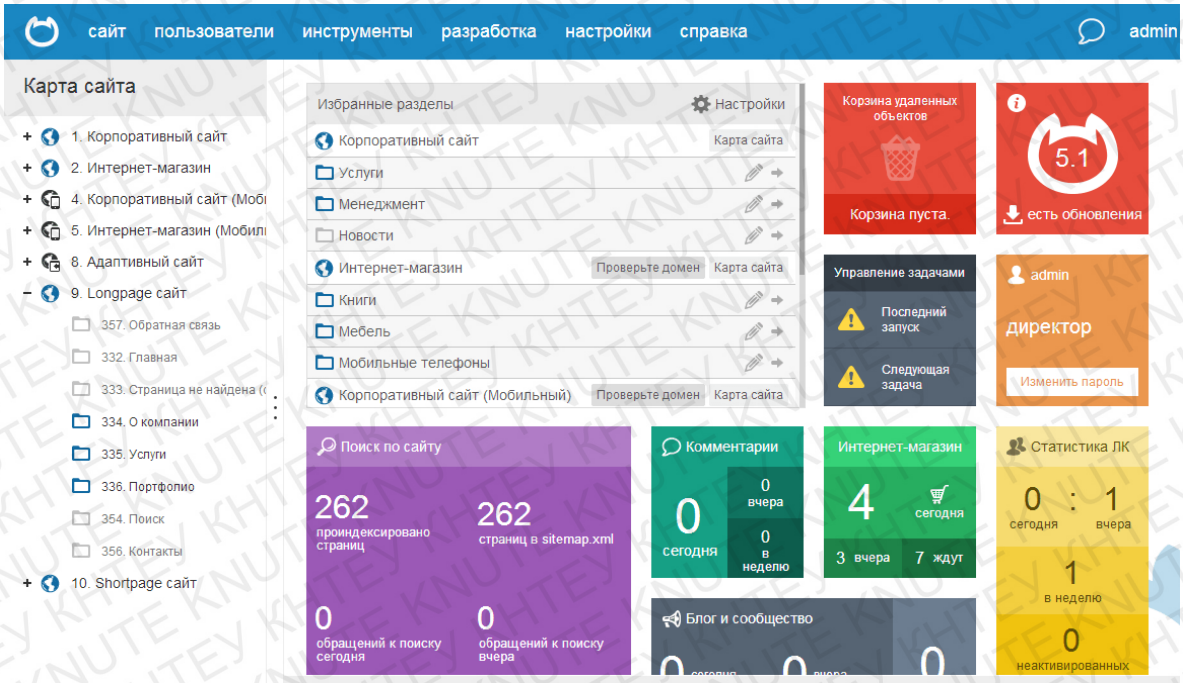


Рис. 2.2. Інтерфейс UMI.CMS

- PrestaShop

CMS з відкритим вихідним кодом яка, завдяки своїй гнучкості, активно використовується при розробці інтернет-магазинів самого різного рівня. На даний момент творцями PrestaShop інтегровано в систему більш ніж 500 функцій

для вирішення найрізноманітніших завдань бізнесу. Практично все необхідне вже є безкоштовно, а нестандартні індивідуальні потреби вирішуються за рахунок сторонніх розширень або додаткових робіт.

Особливості PrestaShop:

- велика кількість вбудованого в систему функціоналу, якого більш ніж достатньо для більшості інтернет-магазинів;
- безліч сучасних шаблонів дизайну, адаптованих під використання на мобільних пристроях, більшість яких дійсно якісні;
- висока швидкість роботи навіть при великому каталозі товарів з десятками тисяч продуктів;
- вартість деяких модулів вище середніх показників, тому в деяких випадках вигідніше замовляти доопрацювання, ніж купувати готове рішення (рис 2.3) [25].

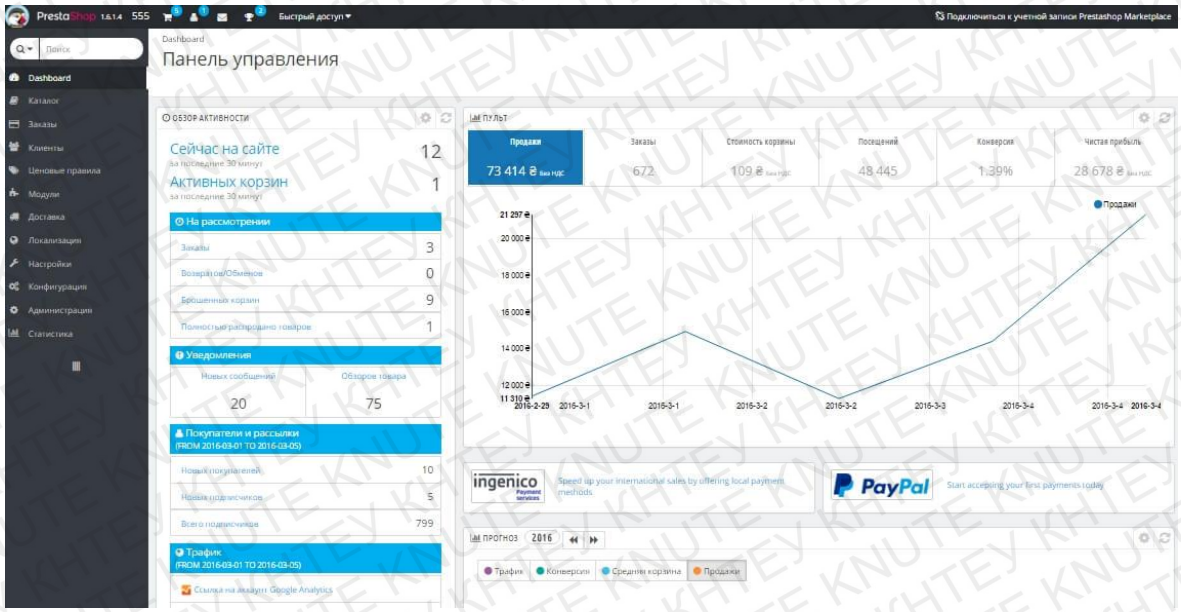


Рис. 2.3. Інтерфейс PrestaShop

- OpenCart

Один з найпростіших в управлінні движків, який користується високою популярністю, посідаючи четверте місце в рейтингах. Основною причиною цього є легкість освоєння адмінпанелі, простота додавання і управління товарами. Недоліки в OpenCart теж присутні. Конфлікти нових версій із застарілими доповненнями, генерація дублів сторінок (негативно позначається

на SEO), обмеженість в плані масштабування. Легкість запуску призводить до витрат, пов'язаних з необхідністю постійних змін і виправлень помилок.

Особливості OpenCart:

- сама система проста, без проблем буде працювати і на віртуальному хостингу;
- шаблони дизайну зазвичай адаптовані під певну версію двигуна, що створює проблеми при їх використанні з виходом нового релізу;
- велика кількість дублів сторінок, які генеруються двигуном, є одним з основних його недоліків (рис. 2.4) [26].

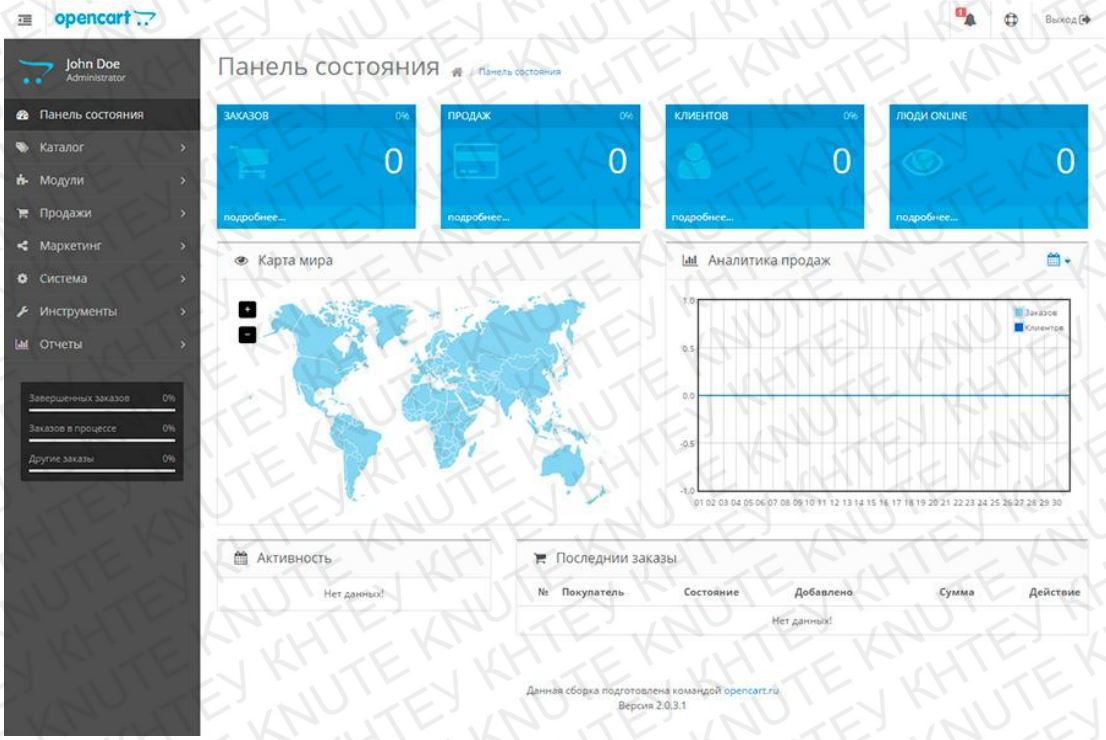


Рис. 2.4. Інтерфейс OpenCart

2.2. Огляд мов програмування web-систем

Основою web-сторінки є розмітка. Технології розмітки, такі як HTML, XHTML і XML, визначають структуру і можливе значення вмісту сторінки. Незважаючи на поширену думку про те, що мови розмітки визначають зовнішній вигляд web-сторінок, і не менш поширене застосування HTML в цьому стилі, зовнішній вигляд сторінки насправді має досягатися за допомогою двох технологій, зокрема, таблиць стилів.

Для створення веб-сайтів використовують такі мови програмування:

- HTML (Hyper Text Markup Language) — мова гіпертекстової розмітки.

Мова розмітки гіпертекстових документів, призначена для створення та написання гіпертекстів. HTML – базується на стандартній мові узагальненої розмітки (SGML - Standard Generalized Markup Language) та призначений для опису взаємопов'язаних документів у розподіленій мережевій інформаційній системі. HTML описує не тільки структуру документів, а й зв'язок між ними. У загальному вигляді HTML – це набір стилів (tags), які виділяють різні компоненти WWW-документів [12].

Кожен файл HTML має однакову базову структуру. Умовно його можна розбити на дві частини – заголовок і тіло. Відповідно є дескриптори, які відносяться до заголовка і тіла html-документу [18] (рис. 2.5):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>HTML Document</title>
  </head>
  <body>
    <p>
      <b>
        Текст буде жирним, <i>Текст курсив</i>.
      </b>
    </p>
  </body>
</html>
```

Рис. 2.5. Структура HTML

Наразі використовується версія HTML5, що є найновішим стандартом HTML. Цей термін представляє дві різні концепції:

- нова версія мови HTML, з новими елементами, атрибутами і поведінкою;
- великий стек технологій, котрі надають більшого різноманіття та потужності Веб-сайтам та додаткам. Цей набір інколи називають HTML5 & friends, але часто скорочують до HTML5.

Спроектована, для використання всіма розробниками Open Web, ця сторінка надає посилання до численних ресурсів про HTML5 технології, розсортовані в декілька груп, в залежності від їх функцій.

Перевагами HTML5 є динамічна складова, яка не потребує використання Java, так як містить в собі базові компоненти цієї мови. Завдяки цьому скорочується час на розробку сторінки та покращується її функціональна частина. У версії HTML5 прописані кодеки для плеєра, що полегшує відображення відеоматеріалу на сторінках сайту та зменшує сам об'єм сторінки. Коли відео не буде працювати мова сама запропонує оновити браузер (табл. 2.1).

Таблиця 2.1

Опис функціональних можливостей HTML 5

Функція	Опис
Семантика	Дозволяє описати якомога точніше з чого складається контент.
Зв'язок	Дозволяє взаємодіяти з сервером новим та інноваційним шляхом.
Offline та сховище	Надає можливість веб-сторінкам зберігати дані на стороні клієнта локально та оперувати offline ефективніше.
Мультимедіа	Робить відео та аудіо першокласними мешканцями Open Web.
2D/3D графіка та ефекти	Дозволяє безліч презентаційних варіантів.
Ефективність та інтеграція	Надає більшу оптимізацію швидкості та кращого використання заліза (hardware) комп'ютера.
Доступ до пристрою	Дозволяє використання різноманітних пристроїв

	вводу/виводу.
Стилі	Дозволяє авторам створювати більш складні і витончені теми.

- XHTML — це нова редакція HTML, виконана за допомогою XML (eXtensible Markup Language, розширювана мова розмітки). XHTML дозволяє вирішити дві основні проблеми, пов'язані з HTML. По-перше, XHTML, приділяючи велику увагу застосуванню таблиць стилів, продовжує чинити тиск на дизайнерів, з тим щоб вони відокремлювали зовнішній вигляд документа від його структури. По-друге, XHTML привносить набагато більш суворі вимоги про дотримання правил розмітки web-сторінок. Наприклад, в документах XHTML повинні міститися тільки теги в нижньому регістрі, атрибути повинні бути обов'язково обрамлені лапками, і, в основному, всі правила в тому вигляді, як вони визначені в специфікації, повинні дотримуватися.

Синтаксична строгість XHTML є одночасно його найбільшим перевагою і найгіршим недоліком. Правильно складеними сторінками може бути простіше управляти й замінювати їх за допомогою програми, але людині їх створювати важче. Перехід на XHTML відбувається повільно саме через його складність. Негнучкість XHTML робить його менш зручним, ніж HTML, який набагато більш поблажливий по відношенню до новачків. Таким чином, поки не з'явиться більша кількість інструментальних засобів, які виконують коректний код XHTML, ймовірно, в масштабах web-спільноти мова буде впроваджуватися так само повільно.

Розширювана мова розмітки (Extensible Markup Language, XML) багатьма сприймається як революційна технологія розмітки, яка змінить вигляд web-сторінок. Проте, незважаючи на цю рекламу, лише деякі розуміють, що насправді таке XML. XML є різновидом SGML, модифікованої для Web. Таким чином, він дозволяє розробникам задавати їх власну мову розмітки.

Негативний сприймається створення занадто великої кількості індивідуальних мов на базі XML, і більшість web-розробників погоджуються користуватися загальноприйнятими мовами XHTML [16, с. 89].

- CSS (Cascading Style Sheets) — каскадні таблиці стилів.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Основна складність роботи з веб-графікою полягає в тому, що пропускна спроможність каналів Інтернету, дуже низька і перед розробником відразу встануть проблеми – як зробити графічний файл невеликий за обсягом, але гарної якості, які програми і прийоми використовувати при його оптимізації. Для цього слід використовувати скрипти, що включають в себе певні налаштування для оптимізації роботи сайту [17, с. 1084] .

CSS має порівняно простий синтаксис і використовує небагато англійських слів для найменування різних складових стилю (рис. 2.6).

Стилі складаються зі списку правил (рис 2.7). Кожне правило має один або більше селектор та блок визначення (рис. 2.8). Блок визначення складається із оточеного фігурними дужками списку властивостей (рис. 2.9).

```
h1, h2 {
  font-family: "Arial", Verdana;
 }
```

Рис. 2.6. Синтаксис у файлі style.css

```
<link rel="stylesheet" href="style.css" type="text/css">
```

Рис. 2.7. Підключення CSS-файлу до сторінки XHTML та HTML

```
<style type="text/css">
@import "style.css";
</style>
```

Рис. 2.8. Імпорт CSS-файлів до HTML та XHTML-сторінок

```
<?xml-stylesheet type="text/css" href="style.css"?>
```

Рис. 2.9. Підключення CSS-файлу до XML-сторінки

- PHP (Hypertext Preprocessor) — препроцесор гіпертексту.

PHP — це мова програмування, що використовується на стороні веб-сервера для генерації і динамічного створення HTML-сторінок. Він створений спеціально для розробки веб-додатків.

PHP — мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег (?>).

Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в C або Pascal.

Наявність інтерфейсів до багатьох баз даних у PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase.

Завдяки стандарту відкритого інтерфейсу зв’язку з базами даних (англ. Open Database Connectivity Standard, ODBC) можна підключатися до всіх баз даних, до яких існує драйвер [19].

Порівняно з подібними мовами програмування PHP має ряд переваг:

- Традиційність.

Мова PHP здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з C, Perl. Код PHP дуже схожий на той, який зустрічається в типових програмах мовами C або Pascal. Це помітно знижує початкові зусилля при вивченні PHP. PHP — мова, що поєднує переваги Perl та C і спеціально спрямована на роботу в Інтернеті, мова з універсальним і

зрозумілим синтаксисом. І хоча PHP є досить молодою мовою, вона здобула таку популярність серед web-програмістів, що в наш час є найпопулярнішою мовою для створення веб-скриптів.

- Наявність сирцевого коду та безкоштовність.

Стратегія Open Source, і розповсюдження початкових текстів програм в масах, безсумнівно справили благотворний вплив на багато проектів, в першу чергу — Linux хоч і успіх проекту Apache сильно підкріпив позиції прихильників Open Source. Сказане відноситься і до історії створення PHP, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проекту PHP. Ухвалення стратегії Open Source і безкоштовне розповсюдження початкових текстів PHP надало неоціненну послугу користувачам. Окрім цього, користувачі PHP в усьому світі є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді, навіть на найскладніші питання.

- Ефективність.

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і web. Важливою перевагою PHP є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте хоч би що робили розробники PHP, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність PHP достатня для створення цілком серйозних веб-скриптів (рис. 2.10).

```
<?php
  echo 'It’s php!';
?>
```

Рис. 2.10. Структура PHP

- JavaScript

JavaScript зазвичай використовується як вбудована мова для програмного доступу до об’єктів додатків. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для надання інтерактивності веб-сторінкам.

Основні архітектурні риси:

- динамічна типізація;
- автоматичне керування пам’яттю;
- прототипне програмування;
- функції як об’єкти першого класу [20].

Послідовність інструкцій (що називається програмою, скриптом або сценарієм) виконується інтерпретатором, вбудованим в звичайний веб-браузер. Код програми вбудовується в HTML - документ і виконується на боці клієнта. Для виконання програми не потрібно перезавантажувати веб-сторінку, всі програми виконуються в відповідь на будь-яку подію. Наприклад, перед відправленням даних форми можна перевірити їх на допустимі значення і, якщо значення не відповідають очікуваним, заборонити відправлення даних (рис. 2.711).

```
<script type="text/javascript">
Alert ('It’s, JavaScript!');
</script>
```

Рис. 2.11. Структура JavaScript

2.3. Вибір методики розробки CMS для голографічної 3D вітрини

Схематично, CMS повинна відповідати наступним вимогам:

- CMS не вимагають професійної технічної підготовки при використовуванні. Практично будь-який користувач може працювати з CMS і управляти вмістом сайту: додавати і видаляти статті, різні модулі (відео, стріми);
- CMS може бути успішно інтегрований у внутрішньокорпоративну інформаційну систему і служити для організації відеострімів;

CMS може бути використаний як інструмент при наладці сайту як каналу взаємостосунків з клієнтами компанії - в цьому випадку доцільно зв'язати CMS і CRM-систему компанії. Ця зв'язка відкриває нові можливості в області підтримки клієнтської "відданості": інформація на сайті мобільна, можливий доступ до окремих, призначеним для певних клієнтів розділом по паролю, що надзвичайно зручно у разі чіткої сегментації клієнтів;

CMS економічно доцільні при організації внутрішньокорпоративних систем і інформаційних порталів [29].

CMS рішень багато, але при виборі варто спиратися на наступні можливості:

- призначений для користувача сервіс – наявність тих або інших функцій і модулів, зрозумілість і доступність користувачу;
- технологічність – використовування технологій, що дозволяють підвищити пропускну спроможність і надійність системи;
- сумісність (апаратна і програмна) – можливість функціонування системи на різних платформах, сумісність з СУБД, можливість підключення додаткових модулів;
- масштабованість – можливість розвитку і нарощування системи.

Насправді, далеко не будь-хто контент-система виявляє собою готовий програмний продукт. Це може бути всього лише набір різнорідних модулів або ж варіант, створений по індивідуальному замовленню. Таким чином, по ступеню готовності контент-системи діляться на наступні різновиди:

- продукт коробочки – готове ПО, дозволяюче встановити систему автоматично і самостійно її набудувати;
- розробник сам встановлює і настроює контент-систему на сервері замовника;
- контент-система проектується і розробляється під кожний окремий проект і встановлюється розробником.

Існує безліч способів економічного обґрунтовування розробки технологічних рішень. Але обрати слід той, який буде максимально швидко

відповідати на запити користувачів та буде максимізувати ефективність при мінімальних витратах на підтримку та експлуатацію [17].

РОЗДІЛ 3

СТВОРЕННЯ CMS ГОЛОГРІФІЧНОЇ 3D ВІТРИНИ

3.1. Розробка алгоритму створення CMS

Умови для ефективної роботи CMS наступні:

- Умова 1. Система управління контентом повинна однаково ефективно працювати на будь-якому пристрої та на будь-якій операційній системі.
- Умова 2. Контент повинен працювати без затримки (або за мінімальною затримкою).
- Умова 3. CMS повинна мати інтуїтивно-зрозумілий інтерфейс.
- Умова 4. Відеоматеріал, що буде відтворюватись в CMS повинен мати можливість зациклюватись. Це потрібно для динамічних презентацій, що дозволить відтворювати матеріал в будь-якому проміжку часу.
- Умова 5. CMS повинна мати можливість контролюватися з серверу, що дозволить відображати контент в будь-якій точці світу одночасно.
- Умова 6. Програма має відтворювати відео з ефектом горизонтальної інверсії зображення.

Для реалізації поставленої мети необхідно обрати простий, але в той же час адаптивний алгоритм функціонування та розробки CMS.
Для реалізації першої умови було обрано мову html. Для реалізації функціональних задач буде використано технологію Java-script. Це зумовлено тим, що відтворення цих двох мов можливе без встановлення тестового серверу та доступу до інтернету. Саме останні 2 пункти вирішують умову 2. Так як тестовий сервер та інтернет знижують продуктивність програми, шляхом завантаження контенту, а не прямого відтворення. Так, як технології html та java-script – це інтернет-технології, то вони також будуть ефективні в онлайні. Саме це вирішує (частково) умову 5.

Умова 3 вирішується шляхом прямого завантаження відеоматеріалу в програму та відтворення її з мініатюри інтуїтиво-зрозумілими кнопками медіаплеєра.

Умова 4 вирішується шляхом додаткового налаштування медіаплеєра, що дозволяє автоматично увімкнути повтор відео без переходу до наступного (навіть, якщо їх завантажено декілька).

Повноцінна реалізація умови 5 відбувається за рахунок стрімів, що запускаються на закритому каналі Youtube. Це дозволя без затримки відтворювати відео на будь-якому пристрої. Для доступу до стріму достатнь у відповідному полі CMS ввести ідентифікатор трансляції.

Умова 6 реалізується шляхом css налаштувань медіаплеєру.

Алгоритм функціонування CMS наступний:

1. В CMS завантажується відеоматеріал.
2. Відбувається запуск веб-додатку.
3. Знаходиться відповідне відео для відтворення.
4. При натискання на кнопку відтворення відбувається відтворювання відео з ефектом повторення.
5. При завершенні перегляду відео на голограмі демонструється чорний екран, що дозволяє приховати від глядачів процес налаштування демонстрації відеоконтенту.

3.2. Розробка web-системи для голографічної 3D вітрини

Розробка CMS починається зі створення движка та підключення плагінів до стандартного каркасу. Після розробки движок має наступний вигляд:

Файл maincontroller_class.php відповідає за аналіз даних на сторінці (рис 3.1)

maincontroller_class.php | abstractcontroller_class.php | route_class.php | view_class.php

```php
<?php

class MainController extends AbstractController {

    protected $title;
    protected $meta_desc;
    protected $meta_key;

    public function __construct() {
        parent::__construct(new View(DIR_TMPL));
    }

    public function action404() {
        parent::action404();
        $this->title = "Страница не найдена - 404";
        $this->meta_desc = "Запрошенная страница не существует.";
        $this->meta_key = "страница не найдена, страница не существует, 404";

        $content = $this->view->render("404", array(), true);

        $this->render($content);
    }

    public function actionIndex() {
        $this->title = "Главная страница";
        $this->meta_desc = "Описание главной страницы.";
        $this->meta_key = "описание, описание главной страницы";

        $content = $this->view->render("index", array(), true);

        $this->render($content);
    }

    public function actionPage() {
```

Рис. 3.1. Файл maincontroller_class.php

Файл abstractcontroller_class.php відповідає за запуск движка (рис 3.2)

maincontroller_class.php | abstractcontroller_class.php | route_class.php | view_class.php

```php
<?php

abstract class AbstractController {

    protected $view;

    public function __construct($view) {
        $this->view = $view;
    }

    abstract protected function render($str);

    public function action404() {
        header("HTTP/1.1 404 Not Found");
        header("Status: 404 Not Found");
    }

}

?>
```

Рис.3.3. Файл abstractcontroller_class.php

Файл route_class.php відповідає підключення до серверу (рис 3.3)

maincontroller_class.php | abstractcontroller_class.php | route_class.php | view_class.php

```php
<?php

class Route {

    public static function start() {
        $controller_name = "Main";
        $action_name = "index";

        $uri = $_SERVER["REQUEST_URI"];
        $uri = substr($uri, 1);
        if ($uri) $action_name = $uri;

        $controller_name = $controller_name."Controller";
        $action_name = "action".$action_name;

        $controller = new $controller_name();
        if (method_exists($controller, $action_name)) $controller->$action_name();
        else $controller->action404();
    }

}

?>
```

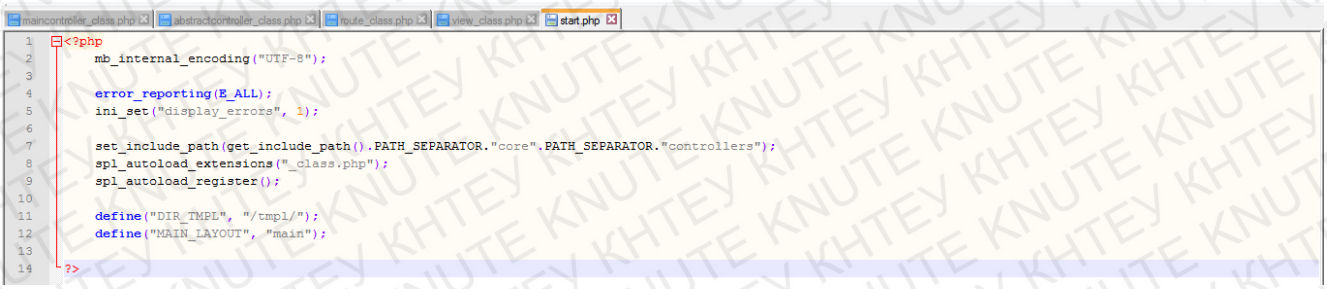
Рис. 3.3. Файл route_class.php

Файл view_class.php відповідає функціонування послідовності запуску скриптів (рис 3.4)

```php
<?php

class View {

    private $dir_tmpl;

    public function __construct($dir_tmpl) {
        $this->dir_tmpl = $dir_tmpl;
    }

    public function render($file, $params, $return = false) {
        $template = $this->dir_tmpl.$file.".tpl";
        extract($params);
        ob_start();
        include($template);
        if ($return) return ob_get_clean();
        else echo ob_get_clean();
    }

}

?>
```

Рис. 3.4. Файл view_class.php

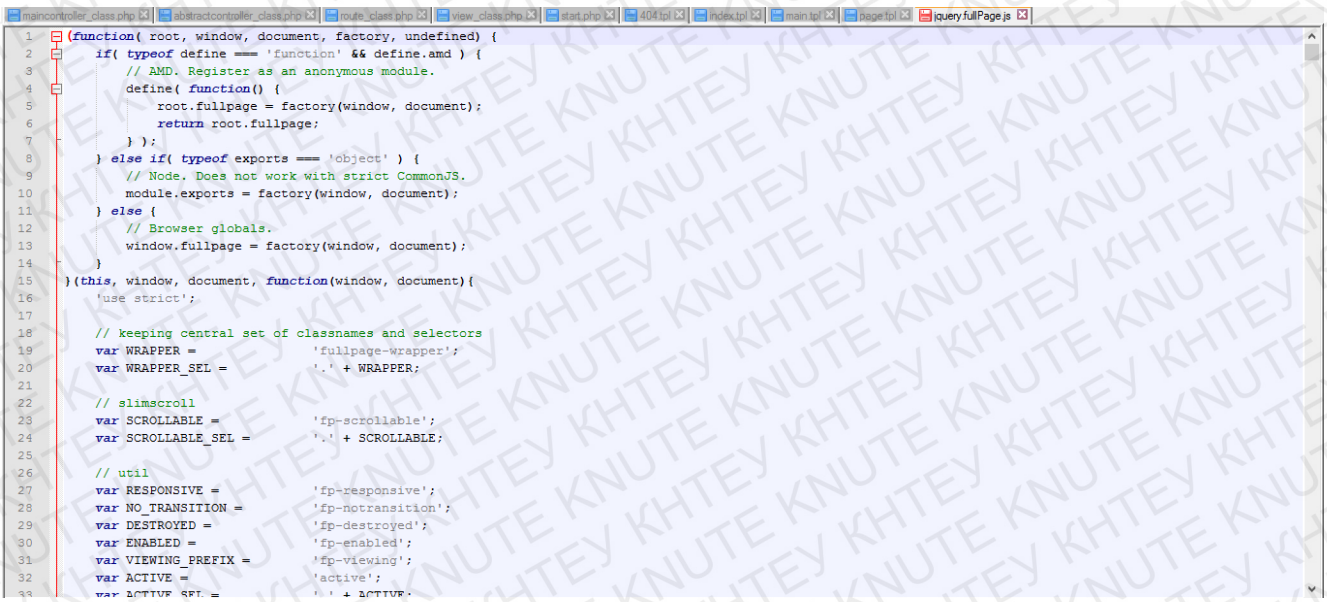
Головною функціональною складовою CMS є серверний файл start.php. Він запускає всі скрипти, імітуючи online сервер.



```php
<?php
    mb_internal_encoding("UTF-8");

    error_reporting(E_ALL);
    ini_set("display_errors", 1);

    set_include_path(get_include_path().PATH_SEPARATOR."core".PATH_SEPARATOR."controllers");
    spl_autoload_extensions("_class.php");
    spl_autoload_register();

    define("DIR_TMPL", "/tmpl/");
    define("MAIN_LAYOUT", "main");

?>
```

Рис. 3.5. Файл start.php

Головним в цій CMS є те, що відеоматеріал відображається в інверсованому вигляді За це відповідає плагін jquery.fullPage.js (рис 3.6) (дод.А)



```javascript
(function( root, window, document, factory, undefined) {
    if( typeof define === 'function' && define.amd ) {
        // AMD. Register as an anonymous module.
        define( function() {
            root.fullpage = factory(window, document);
            return root.fullpage;
        } );
    } else if( typeof exports === 'object' ) {
        // Node. Does not work with strict CommonJS.
        module.exports = factory(window, document);
    } else {
        // Browser globals.
        window.fullpage = factory(window, document);
    }
}(this, window, document, function(window, document){
    'use strict';

    // keeping central set of classnames and selectors
    var WRAPPER =               'fullpage-wrapper';
    var WRAPPER_SEL =           '.' + WRAPPER;

    // slimscroll
    var SCROLLABLE =            'fp-scrollable';
    var SCROLLABLE_SEL =        '.' + SCROLLABLE;

    // util
    var RESPONSIVE =            'fp-responsive';
    var NO_TRANSITION =         'fp-notransition';
    var DESTROYED =             'fp-destroyed';
    var ENABLED =               'fp-enabled';
    var VIEWING_PREFIX =        'fp-viewing';
    var ACTIVE =                'active';
    var ACTIVE_SEL =            '.' + ACTIVE;
```

Рис. 3.6. Плагін jquery.fullPage.js

Основною особливістю фалу css.css є мультиплатформенність, яка досягається шляхом аналізу екрану та підстановки правильних параметрів автоматично (рис 3.7)

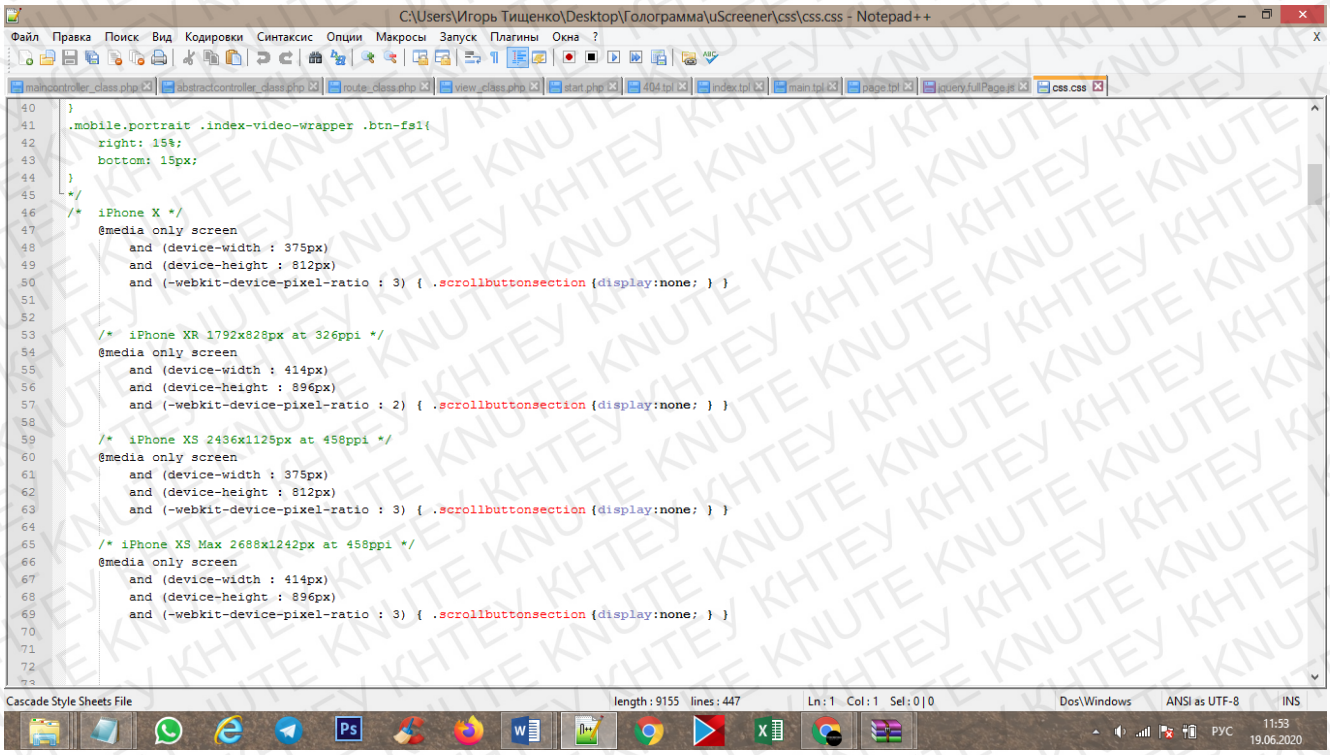


Рис. 3.7. Файл css.css

Основна сторінка має наступний вигляд (рис 3.8)

Рис. 3.8. Файл video.html

Після запуску програмне забезпечення має наступний вигляд (рис 3.9):

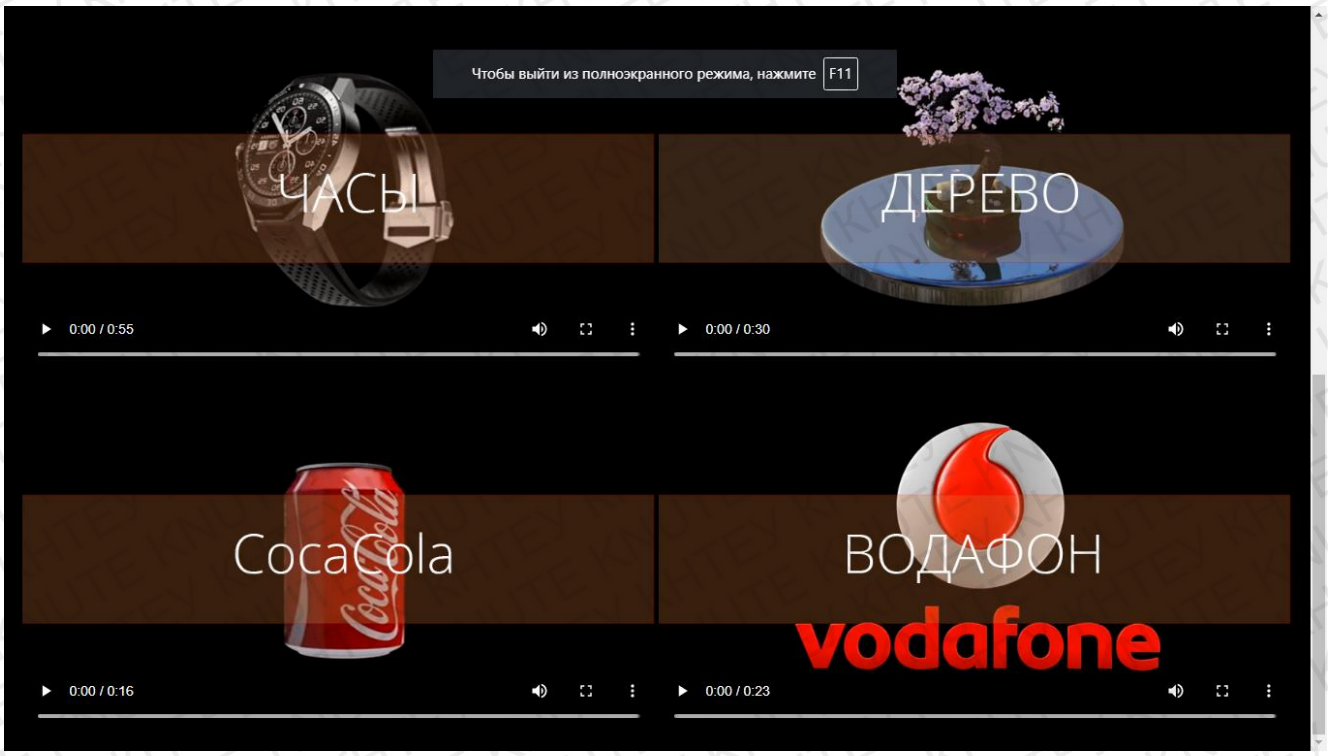


Рис. 3.9. Головна сторінка

Є передперегляд та при кліку на збільшення зображення відео стає повноекранним (рис. 3.10)

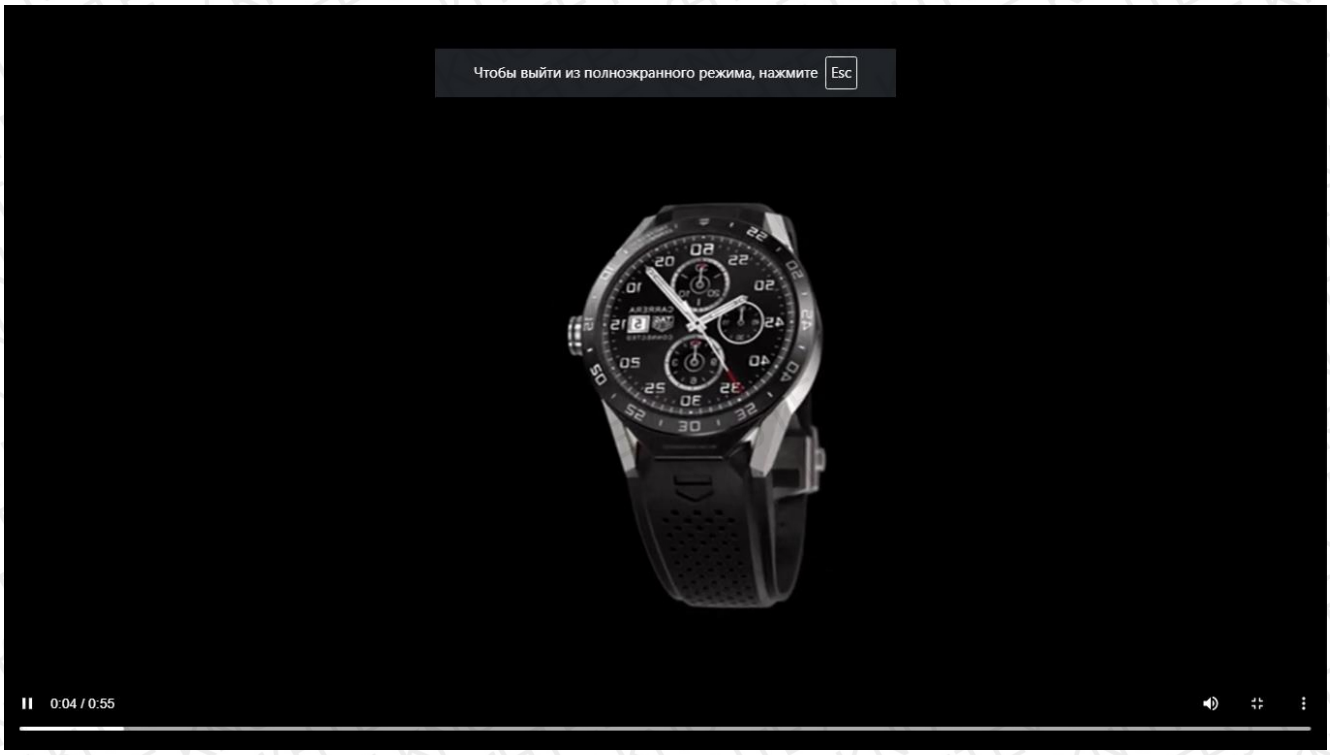


Рис. 3.10. Повноекранне відео

3.3. Підключення CMS

Для доступ до програмного забезпечення online необхідно завантажити його на сервер. За основний сервер обрано сайт розробників голографічної вітрини.

Процес встановлення CMS має наступний вигляд (рис.3.11):

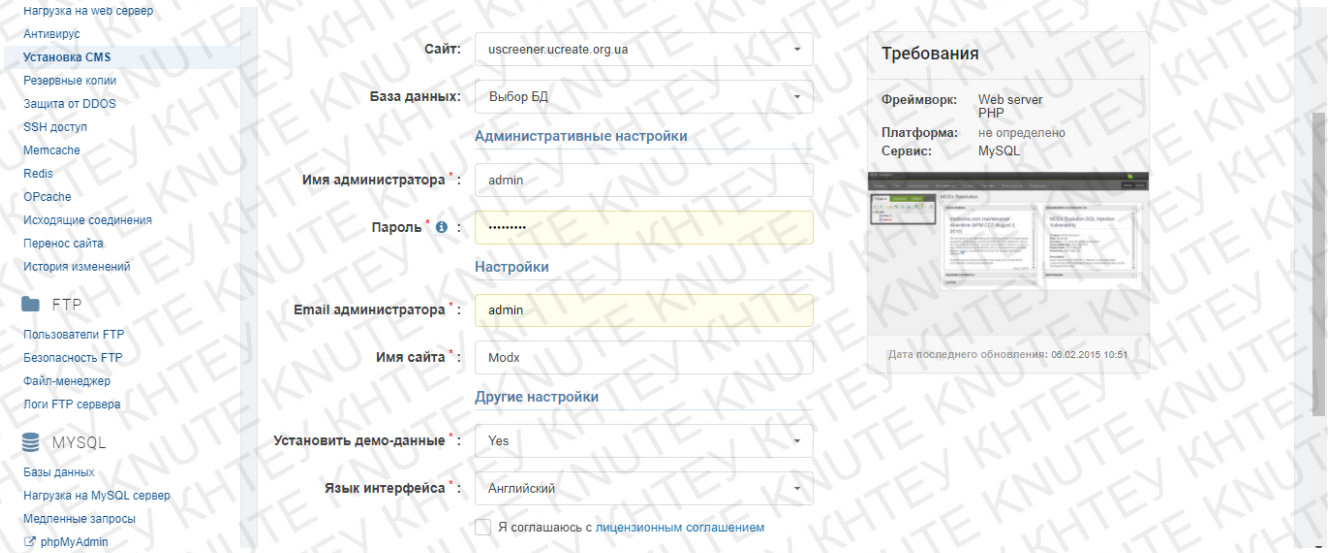


Рис. 3.11. Встановлення CMS

Після встановлення на сервер CMS має наступний вигляд (рис 3.12)

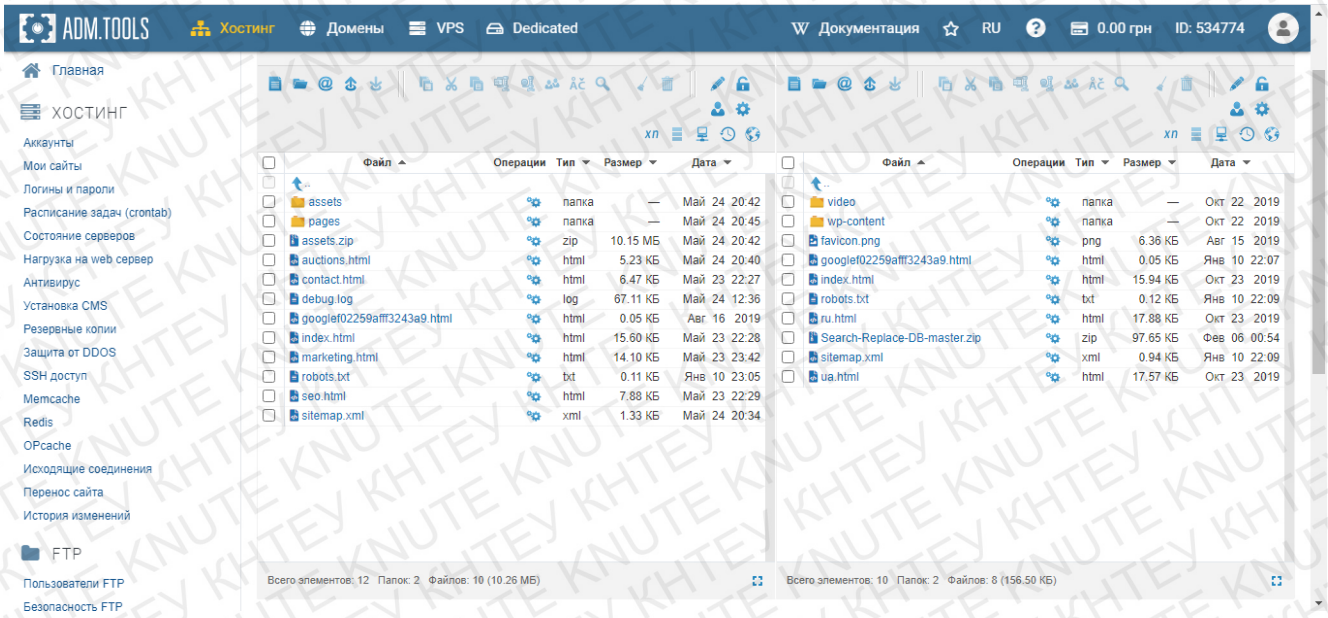


Рис. 3.12. CMS на сервері

Перевірка навантаження на сервер показує, що CMS парцює стабільно. Стабільність роботи перевіряється натсупним чином (рис 3.13, 3.14):

- Сині та зелені відмітки вказують на те, що всі процеси процюють нормально. Перевантажень немає.
- Червоні лінії вказують на те, що деякі процеси працюють не правильно. Тому потрібно або збільшити обсяг оперативну пам’ять сервера або зменьшити кількість функцій.

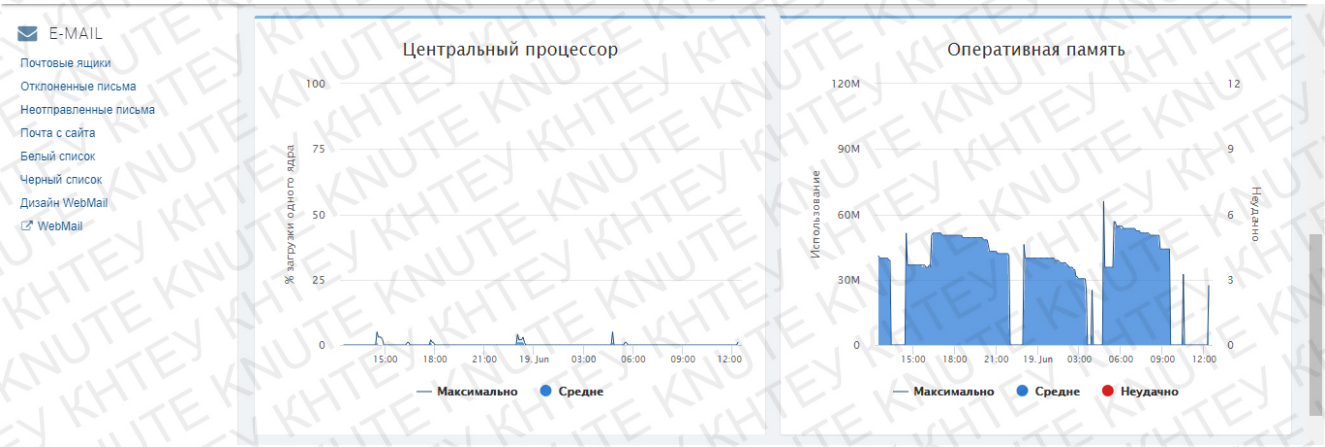


Рис. 3.13. Центаральний процесор та оперативна пам’ять

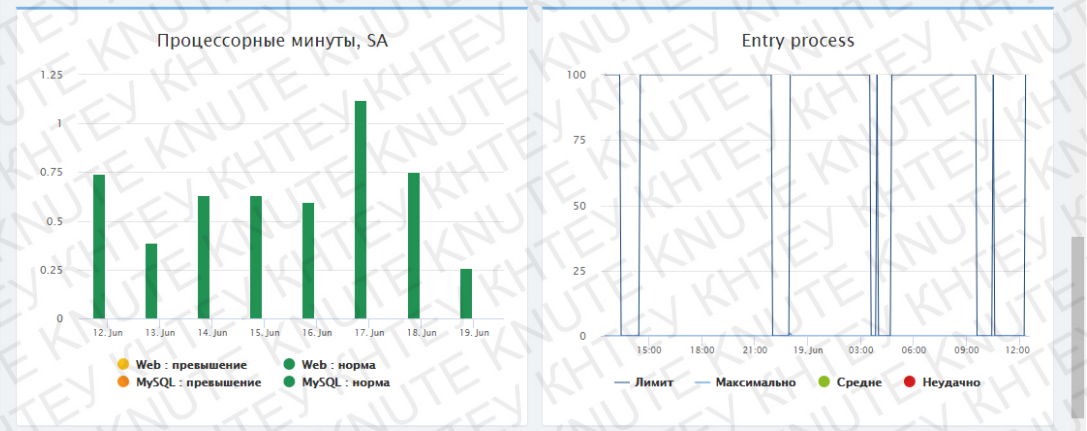


Рис. 3.14. Осоновний процесор та навантаження на сервер

3.4. Правила роботи з CMS

Для запуску веб-додатку необхідно завантажити додаток на комп’ютер та завантажити відео через відповідну форму. Відтворення відео відбувається на сторінці video.html.

Для відтворення відео з серверу необхідно перейти за посиланням в мережі інтернет т, завантажити необхідний матеріал для відтворення через панель

адміністратора та відтворити відео за посиланням:
https://usreener.ucreate.org.ua/video.htm

ВИСНОВКИ

Мета дослідження полягає в практичній реалізація системи управління контентом голографічної 3D вітрини з урахуванням мультиплатформенного користування.

В роботі було проведено аналіз сучасних інформаційних технологій щодо розробки CMS та web сайтів. Використання сучасних технологій дозволяє проводити глибокий аналіз всіх процесів, що вібуваються на сервері моделювати поведінку CMS на різних пристоях з різною швідкістю інтернету.

Для вирішення поставленої в роботі мети було проведено аналіз сучасних методів розробки інформаційних систем підприємства торгівлі та визначено найактуальніші серед них. Провівши аналіз технологій було визначено технології, за допомогою яких найзручніше реалізувати дану CMS.

В другому розділі були досліджені сучасні мови веб-програмування, що дало змогу порівняти структурний рівень кожної з мов, їх пристосованість до реалізації певних завдань. Як висновок, було обрано мову веб-програмування html та JS, що дозволяю створювати веб-сайти різної складності, з використанням мінімуму ресурсів.

В третьому розділі проводився відбувалась практична реалізація алгоритму, CMS та розробка самої CMS.

За результатами розробки отримано алгоритм роботи CMS програмне забезпечення та серверну реалізацію. Висновком роботи є готова CMS, яка висвітлює реальні можливості голографічної 3D вітрини без використання сторонніх програм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Краскевич В.Є., Тищенко І.А. // Математичне моделювання в економіці. – 2019. – № 1.
2. Програмна інженерія : підруч. для студ. вищ. навч. закл. / К.М. Лавріщева. – К. : Київ. УДК, 2008. – 266 с.
3. Інтернет-портали: збірник наукових статей, збірник 2 / А. Н. Тихонов – К. : Київ., 2008. – 499 с.
4. Бази даних в Інтернеті: практичний посібник зі створення Web-додатків з базами даних / А.В. Фролов, Г.В. Фролов, – К. : Київ., 2009. – 89 с.
5. Web-дизайн : навч. посіб., збірник 3 / Т. Пауелл – К. : Вінниця., 2008. - 1084 с.
6. Web-програмування та web-дизайн. Технологія XML : навч. посіб. / О.Б. Проценко. – К. : Суми. СумДУ, 2009. – 127 с.
7. MySQL: лабораторний практикум : нав. книга / Н.Р.Балик, В.І. Мандзюк. - К. : Тернопіль. 2008. - 7 с.
8. Економіко-математичне моделювання соціально-економічних систем: збірник наукових праць МННЦ ІТіС, збірник 14 / Т. П. Подчасова – К. : Київ. УДК, 2009. – 24 с.
9. Системи управління базами даних та знань: книга 2 / 8. А. Ю. Берко, О. М. Верес, В. В. Пасічник. – К. : Львів. «Магнолія-2006», 2012. – 584 с.
10. Економіко-математичне моделювання соціально-економічних систем [Електронний ресурс]. – Режим доступу: http://ua.kursoviks.com.ua/metodychni_vkazivky/article_post/274-v-rtualn-p-dpri-mstva-yak-suchasna-forma-organ-zac-virobnictva
11. Поняття і функції інтернет-магазину. [Електронний ресурс]. – Режим доступу: http://ukrbukva.net/page,6,78319-Saiyt-sportivnogo-kluba-bodibildinga-s-internet-magazinom.html
12. Інформаційні системи. [Електронний ресурс]. – Режим доступу: https://studfiles.net/preview/5725402/page:4

13. Розвиток інформаційних систем: моделі, сутність та стадії. [Електронний ресурс]. – Режим доступу: osvita.ua/vnz/reports/management/13928
14. Проектування інтернет-магазину для підприємства роздрібної торгівлі. [Електронний ресурс]. – Режим доступу: uawsi.com/22/220880.html
15. Методи та засоби створення інформаційної системи [Електронний ресурс]. – Режим доступу: http://helpiks.org/1-27675.html
16. Технології розробки сайтів. [Електронний ресурс]. – Режим доступу: http://web.if.ua/136-tehnologyi-rozrobki-saytv-chastina-1.html
17. Створення Web-сторінок з допомогою мови HTML. [Електронний ресурс]. – Режим доступу: http://programming.in.ua/web-design/html/50--web-html.html
18. Web – програмування. [Електронний ресурс]. – Режим доступу: http://nadoest.com/web--programuvannya-chastina2
19. Програмування сайтів. Бази даних для сайтів. [Електронний ресурс]. – Режим доступу: http://webstudio2u.net/ua/programming/494-site-programming.html
20. Мова JavaScript та її можливості - Web технології та web дизайн. [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/webtehnologiietawebdizajn/mova-javascript-ta-ieie-mozlivosti
21. Керівництво по PHP. [Електронний ресурс]. – Режим доступу: https://php.net/manual/ruPHP
22. Створення сайтів - CMS за і проти. [Електронний ресурс]. – Режим доступу: http://orangestudio.com.ua/all/111-site-development-cms.html
23. Опис CMS 1С-Бітрікс і оцінок. [Електронний ресурс]. – Режим доступу: http://lemarbet.com/ua/otkrytie-internet-magazina/vybiraem-dvizhok-dlya-internet-magazina-kakaya-cms-luchshe
24. Системи управління сайтом. [Електронний ресурс]. – Режим доступу: https://www.bitrix.ua/products/cms/
25. Система управління контентом CMS PrestaShop [Електронний ресурс]. – Режим доступу: www.phpcoders.org.ua › Інтернет технології

26. Обираєм двигун для інтернет-магазину: яка CMS краще. [Електронний ресурс]. – Режим доступу: http://lemarbet.com/otkrytie-internet-magazina/vybiraem-dvizhok-dlya-internet-magazina-kakaya-cms-luchshe
27. Веб-сервери Інтернет — Wiki TNEU. [Електронний ресурс]. – Режим доступу: http://wiki.tneu.edu.ua/index.php?title=1_Веб-сервери_Інтернет
28. Що таке браузер Які браузери популярні на сьогоднішній день. [Електронний ресурс]. – Режим доступу: http://vse-prosto.xn--b1ag1aeig3e.xn--p1ai/shho-take-brauzer-jaki-brauzeri-populjarni-na.html
29. Веб-сервер (Web Server). [Електронний ресурс]. – Режим доступу: http://hi-news.pp.ua/tehnka-tehnologyi/2337-veb-server-web-server-dlya-chogo-vn-potrben-yak-vlashtovano-yak-pracyuye.html

Додаток А

Модуль jquery.fullPage.js

```
(function($) {
 "use strict";
 var defaults = {
  firstClass: 'header', // classname of the first element in your page content
  fullSlideContainer: 'full', // full page elements container for
  singleSlideClass: 'slide', // class for each individual slide
  nextElement: 'div', // set the first element in the first page series.
  previousClass: null, // null when starting at the top. Will be updated based on current postion
  lastClass: 'footer', // last block to scroll to
  slideNumbersContainer: 'slide-numbers', // ID of Slide Numbers
  bodyContainer: 'pageWrapper', // ID of content container
  scrollMode: 'featuredScroll', // Choose scroll mode
  useSlideNumbers: false, // Enable or disable slider
  slideNumbersBorderColor: '#fff', // outside color for slide numbers
  slideNumbersColor: '#000', // interior color when slide numbers inactive
  animationType: 'slow' // animation type: currently doesn't do anything
 };
 $.fn.alton = function(options) {
  var settings = $.extend(true, {}, defaults, options),
   singleSlideClass = settings.singleSlideClass,
   singleSlide,
   bodyScroll,
   down = false,
   up = false,
   current,
   next = $('.' + singleSlideClass + ':first'),
   previous = null,
```

```
  last = $('.' + settings.lastClass),
  projectCount = $('.' + settings.fullSlideContainer).children().length,
  slideNumbers,
  top = true,
  upCount = 0,
  downCount = 0,
  windowHeight = $(window).outerHeight(),
  animating = false,
  docElem = window.document.documentElement,
  scrollOffset,
  offsetTest,
  i;
if ('getElementsByClassName' in document) {
  singleSlide = document.getElementsByClassName(singleSlideClass);
} else {
  singleSlide = document.querySelectorAll('.' + singleSlideClass);
}
if ($('.' + settings.firstClass).length > 0) {
  current = $('.' + settings.firstClass);
} else {
  previous = null;
  current = next;
  next = current.next();
  last = $('.' + singleSlideClass + ':last-child')[0];
}
function initiateLayout(style) {
  if (style === 'featuredScroll') {
    for (i = singleSlide.length - 1; i >= 0; i -= 1) {
      if (is_mobile()) {
        if ($(singleSlide[i]).height() > windowHeight) {
```

```
            $(singleSlide[i]).css('height', $(singleSlide[i]).height());
          } else {
            $(singleSlide[i]).css('height', windowHeight);
            $(singleSlide[i]).outerHeight(windowHeight);
          }
        } else {
          $(singleSlide[i]).css('height', windowHeight);
          $(singleSlide[i]).outerHeight(windowHeight);
        }
      }
      if (settings.useSlideNumbers && !is_mobile()) {
        $('.' + settings.bodyContainer).append('<div id="' + settings.slideNumbersContainer + '"></div>');
        $('#' + settings.slideNumbersContainer).css({
          'height': '100%',
          'position': 'fixed',
          'top': 0,
          'right': '0px',
          'bottom': '0px',
          'width': '86px',
          'z-index': 999
        });
        if (is_mobile()) {
          $('#' + settings.slideNumbersContainer).css({
            'height': 'auto',
            'min-height': '100%'
          });
        }
        $('.' + settings.bodyContainer + ' #' + settings.slideNumbersContainer).append('<ul></ul>');
```

```
        $('.' + settings.bodyContainer + ' #' + settings.slideNumbersContainer + '
ul').css({
         'transform': 'translateY(-50%)',
         '-moz-transform': 'translateY(-50%)',
         '-ms-transform': 'translateY(-50%)',
         '-o-transform': 'translateY(-50%)',
         '-webkit-transform': 'translateY(-50%)',
         'top': '50%',
         'position': 'fixed'
        });
        var testCount = 0;
        while (testCount < projectCount) {
         $('.' + settings.bodyContainer + ' #' + settings.slideNumbersContainer + '
ul').append('<li class="paginate"></ul>');
         if (msieversion()) {
          $('.paginate').css({
           'cursor': 'pointer',
           'border-radius': '50%',
           'list-style': 'none',
           'background': settings.slideNumbersBorderColor,
           'border-color': settings.slideNumbersBorderColor,
           'border-width': '2px',
           'border-style': 'solid',
           'height': '11px',
           'width': '11px',
           'margin': '5px 0'
          });
         } else {
          $('.paginate').css({
           'cursor': 'pointer',
```

```
        'border-radius': '50%',
        'list-style': 'none',
        'background': settings.slideNumbersBorderColor,
        'border-color': settings.slideNumbersBorderColor,
        'border-width': '2px',
        'border-style': 'solid',
        'height': '10px',
        'width': '10px',
        'margin': '5px 0'
       });
     }
     testCount += 1;
    }
    if (('getElementsByClassName' in document)) {
     slideNumbers = document.getElementsByClassName('paginate');
    } else {
     slideNumbers = document.querySelectorAll('.paginate');
    }
   }
  } else {
   $('.' + settings.firstClass).css('height', windowHeight + 10);
   if (!$('.' + settings.firstClass).hasClass('active')) {
    $('.' + settings.firstClass).toggleClass('active');
    if (msieversion()) {
     $('.paginate.active').css({
      'margin-left': '-1px',
      'border-color': '#' + settings.slideNumbersBorderColor,
      'border-style': 'solid',
      'border-width': '2px',
      'height': '8px',
```

```
            'width': '8px'
          });
        }
      }
    }
  }
  function msieversion() {
   var ua = window.navigator.userAgent;
   var msie = ua.indexOf("MSIE ");
   if (msie > 0 || navigator.userAgent.match(/Trident.*rv\:11\./)) { // If Internet Explorer, return version number
    return true;
   } else {
    return false;
   }
  }
  function is_mobile() {
   return navigator.userAgent.match(/(iPhone|iPod|iPad|Android|BlackBerry|BB10|Windows Phone|Tizen|Bada)/);
  }
 or up when called etc.
  function slideIndex(element, toggle) {
   if (toggle && $(slideNumbers[$(element).parent().children().index(element)]).hasClass('active')) {
$(slideNumbers[$(element).parent().children().index(element)]).toggleClass('active');
    $(slideNumbers[$(element).parent().children().index(element)]).css('background', settings.slideNumbersBorderColor);
   } else if (!$(slideNumbers[$(element).parent().children().index(element)]).hasClass('active')) {
```

```
$(slideNumbers[$(element).parent().children().index(element)]).toggleClass('active');
      $(slideNumbers[$(element).parent().children().index(element)]).css('background',
settings.slideNumbersColor);
    }
   }
   function slideNumbersFade(fadeInOut) {
    if (settings.useSlideNumbers) {
     if (fadeInOut) {
      $('#' + settings.slideNumbersContainer).fadeIn();
     } else {
      $('#' + settings.slideNumbersContainer).fadeOut();
     }
    }
   }
   function clickToNavigate(elementIndex) {
    var elementContainer = document.getElementsByClassName(singleSlideClass);
    $(document).scrollTo($(elementContainer[elementIndex]));
    current = elementContainer[elementIndex];
    if ($(current).prev().hasClass(singleSlideClass)) {
     previous = $(current).prev();
    } else {
     previous = $('.' + settings.firstClass);
    }
    if ($(current).next().hasClass(singleSlideClass)) {
     next = $(current).next();
    } else {
     next = $('.' + settings.lastClass);
    }
    slideIndex($('#' + settings.slideNumbersContainer + ' li.active'), true);
```

```
    slideIndex(elementContainer[elementIndex], false);
   }
   function getCurrentPosition() {
    if ($(next).length > 0 || $(previous).length > 0) {
     if ($(window).scrollTop() >= $('.' + singleSlideClass + ':first').offset().top &&
$(window).scrollTop() + $(window).outerHeight() !== $(document).outerHeight()) {
      if (settings.useSlideNumbers) {
       slideIndex(current, true);
      }
      $('.' + singleSlideClass).each(function() {
       offsetTest = $(this).offset().top;
       if (offsetTest <= $(window).scrollTop()) {
        if ($(this).prev().hasClass(singleSlideClass)) {
         previous = $(this).prev();
        } else {
         previous = $('.' + settings.firstClass);
        }
        current = $(this);
        if (current.next().hasClass(singleSlideClass)) {
         next = $(this).next();
        } else {
         next = $('.' + settings.lastClass);
        }
        top = false;
       }
      });
      if (settings.useSlideNumbers) {
       slideIndex(current, false);
      }
      $(document).scrollTo(current);
```

```
    } else {
     if (settings.useSlideNumbers) {
      if (last !== $('.' + singleSlideClass + ':last-child')[0]) {
       slideNumbersFade(false);
      } else {
       slideNumbersFade(true);
       slideIndex(current, false);
      }
     }
     $(document).scrollTo(current);
    }
   }
  }
```