

Детектор Плагиата v. 2961 - Отчёт оригинальности: 21.05.2026 19:40:53

Проанализированный документ: Дипломна Булатова М..docx Лицензия: ВОЛОДИМИР МАТІЄВСЬКИЙ_License2

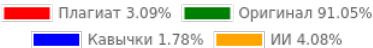
Тип поиска: Поиск переписанного Язык: Uk

Тип проверки: Интернет

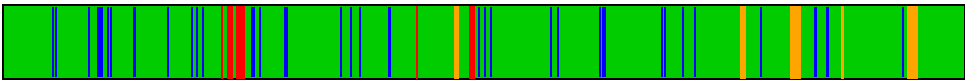
ТЕЕ и кодировка: DocX n/a

Детальный анализ тела документа:

Диаграмма соотношения частей:



Граф распределения зон:



Источники плагиата: 3

→ 2%	2430	1.	https://uk.wikipedia.org/wiki/Система_керування_вмістом
→ 0,7%	735	2.	https://w3schoolsua.github.io/html/index.html
→ 0,2%	200	3.	https://reporter.zp.ua/yak-zrozumiti-proste-chi-skladne-rechennya.html

Детали обработанных ресурсов: 239 - ОК / 30 - Ошибок

Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
Обнаружена Wiki!	[не обнаружено]	[не обнаружено]	Обнаружено соккрытие!

Античит-отчет UACE:

- 1. Статус: Анализатор **Включен** Нормализатор **Включен** сходство символов установлено на **100%**
- 2. Обнаруженный процент загрязнения UniCode: **8,3%** с лимитом: 4%
- 3. Процент нераспознанных символов после нормализации: **4,9%**
- 4. Все подозрительные символы будут отмечены фиолетовым цветом: **Abcd...**
- 5. Найдены невидимые символы: 0

Рекомендации по оценке:
Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется передать это дело на более высокий уровень! В случае сомнений обращайтесь: в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

Исключённые ресурсы:

URL не найдены

Включённые ресурсы:

URL не найдены

Детальний аналіз документа:

ЗМІСТ ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ2 ВСТУП3 РОЗДІЛ 16 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ6
 1.1.Цільові [web](#)-сторінки7 1.1.1.Основні типи цільових сторінок8 1.1.2.Види цільових сторінок9 1.2.Статичні генератори сайтів10 1.3.Системи керування вмістом15
 1.4.Мультисайтинг21 1.5.Шаблон [MVC](#) – головні положення24 1.6.Висновки до розділу 127
 РОЗДІЛ 229 2.1.Мова розмітки [web](#) – документів [HTML](#)31 2.2.Каскадні таблиці стилів – [CSS](#)33
 2.3.Динамічна мова [JavaScript](#)38 2.3.1.[jQuery](#)41 2.4.[Скриптова мова web](#)-сценаріїв PHP42
 2.5.Особливості застосування серверів БД у [web](#)-технологіях45 2.5.1.[MySQL](#)46 2.6.Висновки до розділу 247 РОЗДІЛ 349 РОЗРОБКА ДИНАМІЧНОГО ГЕНЕРАТОРА ВЕБ-САЙТІВ49 3.1.
 Загальний план розробки динамічного генератора веб-сайтів49 3.1.1.Визначення функціоналу динамічного генератора веб-сайтів49 3.1.2.Постановка завдання54 3.2.
 Проектування та розробка основних інструментів ПЗ56 3.3. Розробка програмного коду ПЗ57 3.4. Висновки до розділу 362 ВИСНОВКИ64 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ65
 ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ [CGI Common Gateway Interface](#); [CMS](#) система управління вмістом; [CSS Cascade Style Sheets](#); [DDOS Distributed Denial-of- service attack](#); [DOM Document Object Model](#); [DTD Document Type Definition](#); [EDGE Enhanced Data rates for GSM Evolution](#); [FDA Food and Drug Administration](#); [FSM Finite-state machine](#); [GPRS General Packet Radio Service](#); [HTML HTTP Hyper Text Markup Language](#); [HyperText Transfer Protocol](#); [ISO International Organization for Standardization](#); [JS JavaScript](#); [PHP Hypertext Preprocessor](#); [SEO Search Engine Optimization](#); [SGML SQL UML Standard Generalized Markup Language](#); [Structured Query Language](#); [Unified Modeling Language](#); [URL Uniform Resource Locator](#); [WWW World Wide Web](#); [WYSIWYG What You See Is What You Get](#); [XHTML Extensible Hypertext Markup Language](#); [XML Extensible Markup Language](#); ООП об'єктно-орієнтоване програмування; ОС ПЗ операційна система; програмне забезпечення; ПК персональний комп'ютер. ВСТУП Актуальність теми. Щодня в світі реєструються сотні тисяч нових [Web](#)-ресурсів. Однак такий прогрес тягне за собою збільшення їх якісних показників. Сьогоднішні [Web](#)-системи настільки динамічні, що з їх допомогою здійснюються електронні платежі, робляться інженерні розрахунки і інші типи діяльності коли сутність розгорнутої мережі потребує дуже швидких змін. Це стало можливим завдяки появі технологій формування динамічних [Web](#)-сторінок, вміст яких генерується на стороні серверу і може поряд зі статичними об'єктами включати вибірку з бази даних, результати складних математичних розрахунків і інших складнощів при адмініструванні та оновленні великого масиву файлів [_]. Для цих цілей існує велика кількість інструментів починаючи від сервісів безкоштовного хостингу які дозволяють швидко розгортати сторінки і закінчуючи платними хостингами на яких може встановлюватись спеціалізоване забезпечення для керування даними. Ускладнення функціональності неминує тягне за собою збільшення складності розробки та вимагає від програміста знання кількох технологій програмування. У зв'язку з цим аналіз основних підходів до архітектури серверних додатків, аналіз ефективності застосування базових [web](#)-технологій на стороні сервера, і, отже, вибір технології розробки багаторівневих [web](#)-систем, є актуальним напрямком досліджень. Об'єкт дипломної роботи – технології розробки розподілених [web](#)-систем. Предмет дослідження – принципи ефективної динамічної генерації веб- сайтів, мова програмування [JavaScript](#). Мета роботи – дослідження ефективності базових [web](#)-технологій створення серверних [web](#)-додатків; розробка програмного продукту який буде здатний створювати сайти з динамічним вмістом та функціональністю. Для досягнення мети дослідження необхідно вирішити такі завдання: 1. Провести аналітичний огляд базових технологій розробки [web](#)-систем. 2. Визначити сутність методики динамічної генерації веб-сайтів. 3. Виконати аналіз базових [web](#)-технологій на відповідність критеріям. 4. Розробити інструмент редагування, видалення та розробки шаблонів коду сайту, автоматичної кросбраузерності для властивостей [CSS](#). Наукова новизна: 1. Обґрунтовано доцільність використання серверних мов програмування для проектування [Web](#)-систем на принципах динамічного маніпулювання вмістом; 2. Розроблено конструктор сайту з інструментами обробки коду, який дає можливість зменшити розмір коду сайту та зробити вендорні префікси для всіх властивостей, які їх потребують для створення кросбраузерності. Практичне значення. Результати дослідження можуть застосовуватися для вивчення поняття багаторівневої [Web](#)-системи, алгоритмів роботи основних [web](#)-технологій на стороні сервера. Матеріал роботи дозволить програмістам зробити стратегічний вибір платформи для розробки [web](#)-системи в майбутніх проектах і підвищити ефективність своїх розробок за рахунок розуміння архітектури додатку. Зв'язок роботи з державними програмами, планами науково- дослідних робіт. Результати дипломної роботи можуть бути використані підприємствами, фірмами, розробниками для створення мережі сайтів та масового наповнення їх контентом. Методи дослідження: 1. Методологічну основу дослідження склали положення системного аналізу; 2. Теоретичний аналіз науково-технічної та довідкової літератури за темою магістерської роботи; 3. В якості методу дослідження ефективності вибирається порівняльний аналіз досліджуваних технологій на відповідність критеріям. Особистий внесок складається в: виборі методів досліджень і технологій реалізації; створення програмного продукту, що реалізує механізми динамічного методу генерації веб-сайтів; Апробація роботи. Матеріали роботи доповідалися на V Міжнародної науково-практичної конференції

Цитування: 0,01%

id: 1

«FUNDAMENTAL AND APPLIED RESEARCH IN THE MODERN WORLD»,

Бостон, США у грудні 2026 року. Публікації. Основні матеріали магістерської роботи відображені у статті:

Цитування: 0,01%

id: 2

«Підходи до автоматизованої генерації вебресурсів у реальному часі» []

. Структура дипломної роботи визначена метою і завданнями дослідження та складається зі вступу, трьох розділів, висновків, списку використаних джерел, який складається з 34 бібліографічних посилань, додатків. Обсяг бакалаврської роботи – 76 сторінок. Перший розділ присвячено аналізу методів розробки сайтів за допомогою використання сторінки

сервісів, популярних [CMS](#); створенню статичних веб-сторінок вручну та найбільш сучасних генераторів статичних веб-сторінок. У другому розділі проаналізовано проблему розробки, наявні інструменти, з'ясовано особливості деяких найпопулярніших веб-орієнтованих засобів розробки, досліджено недоліки існуючих інструментів розробки сайтів. У третьому розділі розглядаються етапи створення програмного засобу з простим інтерфейсом, який виконує можливість візуального конструювання сайту, редагування та вставки шаблонів програмного коду. Розглянуто основні етапи створення інструментів застосування та проведено дослідження значимості кожного з них. Додатки містять фрагменти лістинга коду файлів та сертифікат апробації дослідження. РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ Сайтобудова стрімко розвивається, з'являються все нові способи створення і управління [web](#)-сайтами. Це обумовлено тим, що все більша кількість людей створює власні сайти або сайти компаній. Засоби розробки удосконалюються і стають все більш зручними і простими для розуміння. При створенні веб-сайту перед розробником постає головне питання, якими програмними продуктами користуватись для швидкої та ефективної розробки, при цьому необхідно забезпечити зручність впровадження змін вмісту в дані, багатомовність, оптимізованість для пошукових систем тощо. Існує декілька найбільш широко використовуваних прийомів для досягнення поставленої мети: Використання сторонніх сервісів; Використання однієї з популярних [CMS](#); Розробка власної [CMS](#); Створення статичних веб-сторінок вручну; Генератори статичних веб-сторінок. Надамо характеристики кожному з них. При використанні сторонніх сервісів накладаються додаткові витрати на обслуговування системи, що не гарантує повної можливості у збереженні даних при зміні сервісу. Популярні [CMS](#) за свою сутність сприйнятливі перед масовими вразливостями і можуть бути або занадто масштабними для необхідного функціоналу або містити певні хибні базові концепції у своїй архітектурі що будуть заважати цілям веб-сайту. Веб-додатки написані самостійно, також значною мірою вразливі до хакерських атак і часто потребують значних витрат для своєї розробки, часто для вирішення певної задачі написання свого власного продукту є надмірними затратами чи фактором який сповільнить бізнесові цілі. Користування статичними сторінками може допомогти в тому разі якщо нам необхідний дешевий сайт-візитівка чи сторінка презентація, при цьому можливо заощадити на більш дешевих послугах хостингу, в тому числі і за рахунок відмови від використання баз даних. При цих перевагах керування статичними сторінками несе в собі деякі складності, перш за все зв'язування контенту та внесених змін між усіма складовими, які необхідно вручну коригувати після кожного оновлення. У наш час, коли інформація стрімко втрачає свою вартість така кропітка робота може бути успішно автоматизована. Може здатися що ідея статичних веб-сайтів у реальності так званого Інтернету 2.0 здається абсурдною, але ті самі так звані рекламні лендінги, активно використовуються маркетологами з усього світу []. Беручи лендінги як приклад ми маємо розуміти, що для них існують типові шаблони, які наповнюються певним контентом та оптимізованими під пошукові системи ключовими словами. Після публікації лендінгів на певні домени, на них купується Інтернет трафік у вигляді користувачів які в свою чергу виконують оплачувані дії, так звані

Цитування: 0%

id: 3

«конверсії»

принносячи прибутки своїм власникам. Для вирішення таких типів задач гарно підходять статичні генератори веб-сайтів. По перше вони допомагають швидко наповнювати контентом типові шаблони, а по друге на відміну від звичайних статичних веб-сайтів можливо відокремити розробку дизайну від наповнення рекламними текстами та іншим. На мою думку корінний недолік статичних систем генерації веб-сайтів, відсутність гнучкості під час оновлення даних, особливо коли мова йде про великі масштаби задля потреб онлайн-маркетингу. Цільові [web](#)-сторінки Сторінка-вітрина, також лендінг ([Landing page](#)) — веб-сторінка, основним завданням якої є збір контактних даних цільової аудиторії. Використовується для посилення ефективності реклами, збільшення аудиторії. Цільова сторінка зазвичай містить інформацію про товар або послугу, це завершальна сторінка воронки продаж, відкривається при натисканні на рекламне оголошення чи ланку (лінк).

Цитування: 0%

id: 4

«Цільова сторінка»

є логічним продовженням рекламного оголошення або посилання. Часто

Цитування: 0%

id: 5

«лендінги»

пов'язані з соціальними медіа, розсилками електронною поштою або маркетинговими кампаніями пошукових двигунів (контекстною рекламою) з метою підвищення ефективності реклами.

Цитування: 0%

id: 6

«Лендінг»

може бути будь-якою сторінкою сайту або спеціально створеною окремою сторінкою. Загальна мета

Цитування: 0%

id: 7

«лендінгу»

перетворення (конверсії) відвідувачів сайту в потенційних покупців, тому її ще часто називають

Цитування: 0%

id: 8

«приманка для клієнтів» []

. В залежності від маркетингової стратегії, процес перетворення, передбачає виконання користувачем певних дій, як-то: оформити замовлення на покупку, продати конкретний продукт в конкретній ситуації (розпродаж, промоакція), залишити контактну інформацію (зазвичай: адреса електронної пошти або номер телефону) підписатися на розсилку,

реєстрація на сайті. інформація для інших користувачів / оголошення про цільову сторінку. віра в бренд — витратити достатньо часу на сайті, дивитися фільми, читати листівки завантажити або встановити якусь програму Відсоток конверсії відвідувачів є показником ефективності сторінки- вітрини, в залежності від галузі він коливається в межах 2-3 відсотків. Це означає, що, наприклад, із 1000 осіб які відвідали

” Цитування: 0%

id: 9

«лендінг»

- 20-30 зробили якусь із вищевказаних дій []. Існує чотири основних типи цільових сторінок (автономна цільова сторінка; мікросайт; головний сайт; сторінка сегментації) та три види (рекламні цільові сторінки; цільові лід-сторінки;

” Цитування: 0%

id: 10

«Вірусні»

цільові сторінки). Основні типи цільових сторінок Автономна цільова сторінка Представляє собою розгорнуту рекламну пропозицію. Основним завданням якої є спонукання користувача до дії прямо зараз - покупка, підписки на послуги або новини компанії, завантаження пробної версії програмного забезпечення і т. ін. Для посилення ефекту використовуються спонукаючі і закликаючі до дії слогани, яскраві і великі кнопки, мінімалістичний дизайн, акценти на основних перевагах пропозиції, таймери зворотного відліку []. Мікросайт Це окремий сайт з інформацією про рекламу послугу або товар, що складається з декількох сторінок (найчастіше не більше 5). Зазвичай Мікро містять мінімум текстової інформації, велику кількість привабливих і гарних зображень товару або послуги, відеоролики. Подібний тип цільових сторінок часто використовується великими компаніями для реклами окремих проектів. Головний сайт В якості цільової сторінки використовується одна або кілька сторінок основного сайту. Така технологія має досить низьку ефективність, так як в дизайні основного сайту багато відволікаючих увагу елементів навігації. Але дозволяє поєднувати вищу конверсію, в порівнянні зі звичайною сторінкою, плюс, зручність для [SEO](#), порівнянню зі звичайним сайтом []. Сторінка сегментації Розробка посадкової сторінки на базі вже існуючого домену, яка складається повністю з автономних цільових сторінок. Має дуже високою конверсією, але, на відміну від всіх інших видів, незручна для [SEO](#) оптимізації. Види цільових сторінок Рекламні цільові сторінки Вид цільових сторінок, який містить велику кількість текстової, відео, графічної інформації про товар (або послуги). Подібні сторінки працюють за рахунок розгубленості відвідувача в потужному потоці інформації. Купівля найчастіше відбувається для виправдання користувачем зусиль і часу, витрачених на сайті []. Цільові лід-сторінки [Lead Capture](#) - сторінки, створені для збору інформації про цільову аудиторію товару або послуги. Використовуються при розробці маркетингової стратегії для скорочення ризиків в області продажів. На основі зібраної про відвідувача інформації формують пропозицію, орієнтоване на конкретного споживача. Лід-сторінки зазвичай містять коротку інформацію про пропозицію і форму опитувальника (зазвичай з мінімальною кількістю полів). Найчастіше для мотивації відвідувачів до заповнення ліда служать заклики залишити свої дані зараз в обмін на вигідні умови і подарунки від компанії [].

” Цитування: 0%

id: 11

«Вірусні»

цільові сторінки Даний вид цільових сторінок не містить відкритої реклами продукту. На сторінці в будь-якій частині ненав'язливо розташовується логотип або слоган компанії.

” Цитування: 0%

id: 12

«Вірусна»

цільова сторінка сприймається як Інтерактивна розвага або дозвілля. Активно поширюється користувачами через соціальні мережі, електронну пошту, чати і т. ін. Реклама маскується під статтю, відеоролик, картинку або гру. За допомогою звикання до розваги, представленою на сторінці, користувач підсвідомо переймається довірою до марки []. Статичні генератори сайтів Статичні сайти простіше в обслуговуванні (ніяких баз даних, ніяких серверних сценаріїв) і більш безпечні в цілому, враховуючи, що єдина річ, яка передається на пристрої користувачів - це файли [HTML](#), [CSS](#) і [javascript](#). Щоб зробити деякі типи сайтів, на кшталт блогів і сайтів з документацією, статичними, просто написати код у [HTML](#)-файлах досить складно. Також непросто підтримувати сайти з масивним контентом, особливо якщо потрібно змінити деякі незначні деталі (наприклад, дизайн). Саме в такому випадку на допомогу приходять генератори статичних сайтів. Такі генератори в основному конвертують (або компілюють) пачку різних вихідних файлів в один сайт. Це означає, що контент можна зберігати окремо від коду макета, а вміст сайту, таке як зображення, можна зберігати взагалі в іншому місці. Існує безліч генераторів статичних сайтів, можливо навіть сотні, тож розглянемо деякі найбільш популярні з них []. [Jekyll](#) - це самий широко використовуваний генератор, що включає чудову документацію, величезне співтовариство і відмінну підтримку. Навіть [GitHub](#) пропонує вбудовану підтримку [Jekyll](#) в сервісі [GitHub Pages](#). [Jekyll](#) може похвалитися підтримкою блогів. Створювати статичні блоги на [Jekyll](#) дійсно просто. Для цього потрібні лише базові знання веб-розробки. [Jekyll](#) дозволяє створювати і використовувати різні плагіни, теги, і навіть робити свої власні конвертери для будь-якої мови розмітки, який ви хочете використовувати в [Jekyll](#). Стандартна мова розмітки [Jekyll](#), як і в більшості інших генераторів - [Markdown](#). [Jekyll](#) включає плагіни для компілювання [Less](#), [Stylus](#), генерування хмари тегів, призначених для користувача сторінок для блогів і багато чого іншого. [Jekyll](#) заснований на движку [Liquid Template](#) від [Shopify](#). Він працює повністю на [Ruby](#), тому його легко встановлювати разом з взаємозалежностями, використовуючи [rvm](#) або за допомогою пакувальника. [Jekyll](#) також включає опції міграції, якщо ви хочете перенести щось з [WordPress](#), [Blogger](#) або будь-якого іншого сайту блогів. Він без сумнівів є першим з генераторів сайтів з точки зору кількості користувачів. І він знаходиться на стадії активної розробки []. [Pelican](#) - це генератор статичних сайтів на [Python](#). Він відрізняється розміщенням мультимедійного контенту, виділенням коду (синтаксису), а також простим генеруванням [RSS](#) і [Atom Feeds](#). [Pelican](#) включає непоганий

набір плагінів, які знаходяться в центральному репозиторії [GitHub](#). Він підтримує три формату документів за замовчуванням: [Markdown](#), [reStructuredText](#) і [Ascii Doc](#). [Pelican](#) досить унікальний, оскільки він створений на [Python](#). Він підтримує потужний движок [Jinja Template Engine](#), заснований на пітона, що дозволяє легко створювати красиві теми і шаблони для [Pelican](#). З точки зору підтримки міграції, [Pelican](#) пропонує підтримку [WordPress](#) і [Tumblr](#). Замість стандартних [YAML](#)-файлів конфігурації, в [Pelican](#) для конфігурації і настройки використовується файл `*.py` під назвою `pelicanconf.py` [1]. [Metalsmith](#) пишається тим, що він є генератором статичних сайтів на основі плагінів. Це означає, що вся логіка [Metalsmith](#) здійснюється плагінами. Яка б функція вам не була потрібна, просто додайте потрібний плагін. Величезна кількість пропонованих в [Metalsmith](#) плагінів здатне обійти будь-якого конкурента (напевно, крім [Jekyll](#) і [Docpad](#)). Це означає, що [Metalsmith](#) можна використовувати як щось більше, ніж просто генератор статичних сайтів. Іншими словами

Цитування: 0,01%

id: 13

«Так як все складається з плагінів, бібліотека насправді являє собою просто абстракцію для управління директорією файлів».

В результаті це дає те, що ви легко використовуєте [Metalsmith](#) як інструмент для роботи з проектом, генератор електронних книг, інструмент для побудови технічної документації та ін. (Це всього кілька прикладів, продемонстрованих на сайті [Metalsmith](#)) [1]. [Harp](#) включає вбудовану попередню обробку для [Jade](#), [Markdown](#), [LESS](#), [Sass](#), [CoffeeScript](#), [EJS](#) і [Stylus](#) без будь-яких додаткових налаштувань. Він також дозволяє використовувати макети / часткові таблиці форм з [Jade](#) і [EJS](#), для чого в інших генераторах статичних сайтів потрібен спеціальний плагін [1]. [Harp](#) побудований на [Node.js](#) і може працювати пліч-о-пліч з [Harp Platform](#), що дозволяє створювати веб-сторінки з папки на [Dropbox](#). [Harp](#) також може компілювати сторінки для використання на сторінках [GitHub](#), а також [PhoneGap](#) і [Heroku](#) [1]. [Docpad](#) - це динамічний генератор статичних сайтів. Він пропонує більш великі можливості, ніж інші генератори статичних сайтів, пропонуючи такі функції, як запити бази даних через движок запитів, імпорт сторінок з зовнішніх баз даних, повторне відображення веб-сторінки при кожному запиті. [Docpad](#) володіє вбудованою підтримкою препроцесорів, таких як [CoffeeScript](#), [Stylus](#) і [LESS](#), і використовує плагіни для підтримки двигунів шаблонів, препроцесорів і мов розмітки, тому ви можете вибрати будь-які бажані комбінації, використовуючи необхідний плагін. [Docpad](#) також підтримує імпорт сторінок з зовнішніх джерел, таких як [Tumblr](#), [GitHub](#) і [Dropbox](#), через плагіни. [Docpad](#) - це багатфункціональна платформа з величезною кількістю плагінів і відмінною документацією. Для зв'язку з сервером він використовує [Node.js](#), на якому і побудований [1]. [Hexo](#) - це легка оболонка для статичних блогів, яка відрізняється величезною швидкістю генерування сайтів. [Hexo](#) відмінно підходить для блогерів з великою кількістю контенту, яким потрібен простий генератор статичних сайтів. Він пропонує зручну функцію міграції з інших платформ блогів, таких як [WordPress](#), [Joomla](#), [Jekyll](#), [Octopress](#) і [RSS](#). А що найважливіше в [Hexo](#) - це те, що в ньому можна використовувати більшість плагінів, створених для [Octopress](#) (і, отже, плагіни, створені для [Jekyll](#) з дуже незначними змінами). [Hexo](#) підтримує [Markdown](#), [YAML](#) для створення титульної сторінки і налаштування. І, не втрачаючи своєї характерної швидкості, [Hexo](#) дозволяє розгортатися на таких сайтах як [GitHub](#), [Heroku](#) і [Rsync](#) за допомогою всього однієї команди [1]. [Hugo](#) - це генератор статичних сайтів загального призначення з відмінними універсальними функціями, такими як підтримка шаблонів і компонентів, розбивка на сторінки і

Цитування: 0%

id: 14

«таксономія»,

яка спочатку є унікальною системою категоризації, успадкованої з [Hugo](#). Це означає, що ви можете розбивати пости на класи не тільки на підставі тегів, але також будь-яким іншим способом на ваш розсуд, наприклад за категоріями або серіями, починаючи з титульної сторінки. [Hugo](#) підтримує три типи файлів даних - [YAML](#), [JSON](#) і [TOML](#), і дозволяє вибирати, з чим вам зручніше працювати. Замість плагінів в [Hugo](#) використовуються

Цитування: 0%

id: 15

«короткі коди»,

які дозволяють використовувати багатий контент всередині [Markdown](#). Щоб дізнатися, як це працює, ознайомтеся з цією статтею. [Hugo](#) написаний на мові програмування [Go](#) і пропонує окремі файли установки для різних платформ на своїй сторінці [GitHub](#) [1]. [Brunch](#) швидше спрямований на веб-додатки в [HTML5](#), ніж на блоги та веб-сайти, але все одно це дуже простий у використанні і швидкий генератор статичних сайтів. Він не тільки компілює весь ваш код і скрипти, але також може автоматично зменшувати (мінімізувати) ваш код і стискати зображення. [Brunch](#) включає цілий ряд плагінів, які ви можете використовувати для настройки генератора відповідно до своїх потреб. [Brunch](#) пропонує

Цитування: 0%

id: 16

«скелети»,

які в основному є шаблонами для створення сайту (або веб-додатки). Він забезпечує мало не найвищу швидкість компіляції, просто тому що [Brunch](#) кешує всі незмінні частини вашого проекту і компілює тільки ті файли, які змінилися. [Brunch](#) побудований на [Node.js](#) [1]. У таблиці 1.1 наведено підсумки проведеного огляду найбільш популярних генераторів, показано мову розробки програмного забезпечення, наявність ліцензії, шаблонів, плагінів, документації, регулярних оновлень та підтримки, вказані особливості інсталяції, підтримки розширення та деплою: Таблиця 1.1. - Аналіз доступності і функціональних можливостей найпопулярніших генераторів сайтів Генератор Мова Ліцензія Інсталяція Підтримка Розширення Деплой [Jekyll](#) [Ruby](#) MIT Потрібна остання версія [Ruby](#), [RubyGems](#), [Rbenv](#) Детальна документація, оновлення виходять регулярно Багато різних розширень і плагінів [Git](#), [FTP](#), [SFTP](#), [rsync](#), [Amazon S3](#), [Heroku](#) [Pelican](#) [Python](#) GNU GPL Встановлюється через [pip](#) Детальна документація, активно розробляється підтримується Багато різних плагінів для блогів [FTP](#), [SSH](#), [Dropbox](#), [Amazon S3](#), [Rackspace](#) [Cloudfiles](#) [Metalsmith](#) [JS](#) MIT

Встановлюється через менеджер пакетів Детальна документація з відеороліками Логіка обробляється плагінами. зв'язує їх ланцюгом [Dropbox](#), [Google Cloud Storage](#), [Amazon S3](#) [Google AppEngine](#) [Harp JS MIT](#) Встановлюється через менеджер пакетів Часткова підтримка [HTML / CSS / JS EJS](#), [Jade](#), [Markdown](#), [Less](#), [Stylus](#), [Sass](#) та [CoffeeScript](#) [DocPad](#) [CoffeeScript MIT](#) Для інсталяції потрібні [NodeJS](#) і [NPM](#) Детальна документація, оновлення виходять регулярно Багато різних розширень і плагінів [Heroku](#), [Appfog](#), [Windows Azure](#), [Docker](#), [GitHub Pages](#) [Hexo JS MIT](#) Для інсталяції потрібні [NodeJS](#) і [NPM](#). Детальна документація, оновлення виходять регулярно Багато різних плагінів для блогінгу [EJS](#), Мопс, [Haml](#), [Swig](#), [Nunjucks](#), Вуса, [Handlebars](#), [Twig](#), [Marko](#) [Hugo](#) [Go Apache](#)-2.0 часткова підтримка використовує файли [Markdown](#) [Google Cloud Storage](#), [Amazon S3](#) [Dropbox](#), [Google AppEngine](#) [Brunch JS MIT](#) Для інсталяції потрібні [NodeJS](#) Детальна документація, оновлення виходять регулярно Багато різних плагінів для блогінгу [Nunjucks](#): [Brunch-Nunjucks-Static Git](#), [FTP](#), [SFTP](#), будь-який [JS](#) Системи керування вмістом Система керування вмістом ([Content Management System](#), [CMS](#))

 **Обнаружен Плагиат: 0,03%** id: 17
https://uk.wikipedia.org/wiki/Система_керування_вмістом
 програмне забезпечення для організації веб-сайтів чи інших інформаційних ресурсів в Інтернеті чи окремих комп'ютерних мережах. Існують сотні, а може, навіть й тисячі доступних [CMS](#) — систем. Завдяки їх функціональності ці системи можна використовувати в різних компаніях

. Незважаючи на широкий вибір інструментальних та технічних засобів, наявних в [CMS](#), існують загальні для більшості типів систем характеристики [1]. Розрізняють наступні типи [CMS](#) [1]: [Web content management systems](#) для управління веб-сайтами (наприклад, енциклопедіями, онлайн-виданнями, блогами, форумами, корпоративними чи персональними веб-сторінками та ін.)

 **Обнаружен Плагиат: 0,1%** id: 18
https://uk.wikipedia.org/wiki/Система_керування_вмістом
 Транзакційні [CMS](#) для забезпечення транзакції у електронній комерції. Інтегровані [CMS](#) для роботи з документацією на підприємствах. Електронні бібліотеки ([Digital Asset Management](#)) для забезпечення циклу життя файлів електронних медіа (відео, графіки, презентації тощо). Системи для забезпечення циклу життя документації (інструкції, довідники, описи). Освітні [CMS](#) — системи для організації Інтернет курсів та відповідного циклу життя документації Платформенні [CMS](#) ([Platform Content Management Systems](#)) підтримують автоматизацію роботи з комп'ютерними файлами, папками, програмами у визначеному програмному середовищі. Корпоративні [CMS](#) ([Enterprise content management systems](#)) з різноплановим пристосуванням для потреб підприємницької діяльності. Підтримують цикл життя внутрішньої і зовнішньої документації. Також існують різні типи роботи [CMS](#) [1]: Генерація сторінок за запитом. Системи такого типу працюють на основі зв'язки

 **Цитування: 0,01%** id: 19
 «модуль редагування - база даних - модуль представлення».

 **Обнаружен Плагиат: 0,16%** id: 20
https://uk.wikipedia.org/wiki/Система_керування_вмістом
 Модуль представлення генерує сторінку з контентом при запиті на нього на основі інформації з бази даних. Інформація в БД змінюється за допомогою модуля редагування. Сторінки заново створюються сервером при кожному запиті, а це створює навантаження на сервер. Але це навантаження може бути багатократно зменшене при використанні методів кешування, які є в сучасних веб-серверах. Генерація сторінок при редагуванні. Системи цього типу при редагуванні сторінок вносять зміну у вміст сайту та створюють набір статичних сторінок. При такому способі втрачається інтерактивність між відвідувачами сайтів та контентом даного сайту. Змішаний тип. Як зрозуміло із назви, цей тип поєднує в собі переваги перших двох. Може бути реалізований шляхом кешування — модуль представлення генерує сторінку один раз, надалі вона через деякий час буде в декілька разів швидше завантажуватися із кешу. Кеш може оновлюватись як автоматично, через деякий час чи при внесенні змін у певні розділи сайту, так і вручну за команду адміністратора. Другий підхід — збереження певних інформаційних блоків на етапі редагування сайту і збирання сторінок з цих блоків при запиті відповідної сторінки користувачем

. Одним з перших конструкторів сайтів, де не потрібно було використовувати спеціальні навички програмування, був проект [Geocities](#), заснований у 1994 році. Після свого 5-річного існування, [Geocities](#) був проданий компанії [Yahoo!](#) за \$ 3,6 млн. Після того, як проект технічно застарів, він був закритий у квітні 2009 року. З того часу ринок конструкторів веб-сайтів представлений більш, ніж сотнею платформами, що дозволяють створювати сайти найрізноманітніших типів [1]. Найбільшу популярність за версією

 **Цитування: 0%** id: 21
 «Рейтингу Рунету»

мають такі [CMS](#): 1С-Битрикс, [Drupal](#), [Joomla!](#), [MODX](#), [UMI.CMS](#), [WordPress](#), [NetCat](#), [HostCMS](#), [CS-Cart](#), [AMIRO.CMS](#), [NespiCMS](#), [Lekast](#).

 **Цитування: 0,03%** id: 22
 «Ви можете самостійно створити як особистий блог (інтернет-щоденник), так і складний інформаційний портал. За допомогою [CMS](#) можна зробити як невеликий сайт-візитку своєї компанії, так і повноцінний корпоративний портал» [1]

. Для розуміння різниці між статичними генераторами веб-сайтів та [CMS](#) розглянемо особливості найбільш популярних. [Joomla!](#) – важливою особливістю системи є мінімальний набір інструментів при початковій установці, який доповнюється в міру необхідності [1]. Функціональність можна збільшувати за допомогою додаткових розширень (компонентів, модулів і плагінів). Є модуль безпеки для багаторівневої аутентифікації користувачів та адміністраторів (використовується власний алгоритм аутентифікації і

 **Цитування: 0%** id: 23
 «ведення»

сесій). Система шаблонів дозволяє легко змінювати зовнішній вигляд сайту або створити свій унікальний. У мережі існує величезний вибір готових шаблонів, як платних, так і безкоштовних. Передбачені схеми розташування модулів, включаючи лівий, правий, центральний і будь-яке інше довільне положення блоку. При бажанні вміст модуля можна включити в вміст матеріалу. Вбудована багатомовність. Розширена підтримка баз даних. Реалізована підтримка [Microsoft SQL Server](#), а з версії 3.0 – [PostgreSQL](#). [Drupal](#) – популярна вільна модульна система керування вмістом з відкритим сирцевим кодом, написана на мові програмування [PHP](#). [Drupal](#) може працювати у таких популярних системах як [Windows](#), [Mac OS X](#), [Linux](#), власне, на будь-якій платформі, яка підтримує роботу веб-сервера [Apache](#), [Nginx](#), [Lighttpd](#) або [Microsoft IIS](#); також потрібна наявність системи керування базами даних [MySQL/MariaDB](#), [PostgreSQL](#) 8.3, [SQLite](#) чи інші комерційні []. У дистрибутиві системи входить набір модулів, що дають наступні можливості: збір інформаційних стрічок ([RSS](#), [RDF](#), [Atom](#)); ведення блогів, підшивань і форумів; створення форм для відправки повідомлень; локалізація системи; перейменування посилань (призначення посиланням зрозумілих і зручних псевдонімів); проведення опитувань; призначені для користувача профілі, що налаштовуються; пошук за змістом (за зміст вважається і повідомлення на форумах, і сторінки, і будь-які інші призначені елементи); ведення журналу статистики (відвідуваності); таксономія (впорядковування матеріалу за категоріями) формування сторінок з матеріалами в різних формах і форматах подання та інші []. [WordPress](#) – проста у встановленні та використанні система керування вмістом з відкритим кодом, яка широко використовується для створення веб- сайтів. Сфера застосування — від блогів до складних веб-сайтів. Вбудована система тем і плагінів в поєднанні з вдалою архітектурою дозволяє конструювати на основі [WordPress](#) практично будь-які веб-проекти. Написана мовою програмування [PHP](#) з використанням бази даних [MySQL](#). Сирцевий код поширюється на умовах ліцензії [GNU General Public License](#). Окрім цього Розробники [WordPress](#) дали можливість користувачам створювати власні плагіни. Всі файли плагінів розміщуються в теці [wp-content/plugins](#). Його головний файл повинен бути написаний на мові [PHP](#). Плагін також може складатись з декількох файлів, якщо вони під'єднані до головного файлу (наприклад за допомогою функції [include](#)). Якщо ж до нього треба приєднати [CSS](#), [JavaScript](#) або інші зовнішні файли []. Дизайн, управління системою та інші можливості: простота встановлення, простота налаштувань; підтримка веб-стандартів ([XHTML](#), [CSS](#)); модуль для підключення (плагіни) з унікально простою системою їх взаємодії з кодом; можливість автоматичного встановлення та оновлення версії безпосередньо з панелі адміністратора; підтримка так званих

” Цитування: 0% id: 24

«ТЕМ»,

з допомогою яких легко змінюється як зовнішній вигляд, так і способи виведення даних; можливість редагувати шаблони одразу в панелі адміністратора;

” Цитування: 0% id: 25

«ТЕМИ»

реалізовані як набори файлів-шаблонів на [PHP](#) (у [HTML](#)- розмітку вставляються [PHP](#)-мітки); багато бібліотек

” Цитування: 0% id: 26

«ТЕМ»

i

” Цитування: 0% id: 27

«плагінів»

; потенціал архітектури дозволяє легко реалізовувати складні рішення; [SEO](#)-оптимізована система; наявність українського перекладу. Публікація та редагування: миттєва публікація; підтримка [RSS](#), [Atom](#), [trackback](#), [pingback](#); наявність ЛЗУ (людино-зрозумілий [URL](#)); редагування [WYSIWYG](#)-редактором з можливістю вставлення форматованого тексту (наприклад з програми [Microsoft Word](#)) або редагування за допомогою [HTML](#)-розмітки. Контент: наперед заплановані публікації; багатосторінкові записи; прикріплення файлів та зображень до записів; можливість створення статичних сторінок; можливість створення свого типу контенту у власних темах; категорії, теги, коментування тощо []. [Wix.com](#) – міжнародна хмарна платформа, написана на [Scala](#) [], для створення і розвитку інтернет-проектів, яка дозволяє конструювати сайти і їх мобільні версії на [HTML5](#) з допомогою інструментів [drag-and-drop](#) []. Розширювати функціональність сайтів можна за рахунок додатків, розроблених [Wix](#) або сторонніми компаніями. Наприклад, додавати плагіни соціальних мереж, інструменти для онлайн-торгівлі і електронних розсилок, контактні форми, блоги та ін []. [Wix](#) працює на бізнес-моделі [freemium](#), пропонуючи можливість створювати сайти безкоштовно і розвивати їх, купуючи корисні поліпшення. Наприклад, тарифи [Premium](#) дозволяють підключити до сайту власний домен, прибрати банери [Wix](#), додати онлайн-магазин, отримати додаткове місце для зберігання даних, купони на рекламу та ін Провівши дослідження можливостей найпопулярніших [CMS](#) конструкторів винесемо їх особливості у таблицю 1.2. та проведемо порівняння їх за функціональністю, доступністю, виділимо переваги й недоліки кожного програмного забезпечення. Таблиця 1.2. - Аналіз Доступності і Функціональних можливостей найпопулярніших конструкторів сайтів Назва Доступність Переваги Недоліки [Joomla!](#) безкоштовна -кожен день виходять нові розширення, модулі і плагіни, які не дозволять сайту застаріти; -достатній функціонал, для створення серйозного проекту з розширеним функціоналом безпеки сайтів на досить високому рівні. -велике навантаження на сервер, відсутність техпідтримки; - слабо розвинена система перетворення [URL](#) створює серйозні труднощі у процесі створення сайтів. [Drupal](#) умовно безкоштовна -не потрібні навички програмування; -велика кількість функціональних модулів: новини, блог, фотогалерея, каталог файлів, опитування, форум, налаштування [SEO](#)-параметрів тощо; -гнучкі налаштування для управління технічними параметрами сайту, виведення контенту, копіювання; -ергономічна, логічна і зручна адміністративна панель; -велика кількість безкоштовних шаблонів для

найрізноманітніших тематик; -можливості доступу для використання на безкоштовному акаунті; -можливість прикріпити домен другого рівня; -багатомовність; -сервери системи витримують практично будь-яке навантаження; -повна сумісність з мобільними пристроями; помірні цінові політики адміністрації. -важко піддається освоєнню і за рахунок свого складного інтерфейсу, який не сприймається на інтуїтивному рівні; -високі вимоги системи; -відсутність у вільному доступі шаблонів з красивим дизайном; -складність установки оновлень і програмних модулів. [WordPress](#) умовно безкоштовна -багато плагінів; -регульований і простий; -часті оновлення; відкрите джерело. -обмежені можливості налаштувань; правки внесені у код, зникають після поновлення ядра [GMS](#) або теми; слаба безпека. [Wix](#) платна (ціни вищі за середньоринкові) -зручна в освоєнні панель адміністрування; сотні красивих і якісних шаблонів майже для будь-якої тематики; -маркетплейс додатків, дозволяє реалізувати інтеграцію з іншими популярними сервісами; -детальна інтерактивна довідкова система. -сайти не адаптивні; -можливість підключення веб-аналітики і [SEO](#)-параметрів; -доступна тільки в преміум-тарифах; -найдорожчий тарифний план. Вказані особливості перелічених [CMS](#) дозволяють їм займати значне положення на ринку всіх встановлених веб-сайтів в мережі, та задля спрощення роботи з веб-сайтами та підвищення рівня безпеки вони не завжди можуть бути використані. Мультисайтинг Мультисайтинг (багатосайтовість) можна визначити як можливість використовувати файли скрипту для різних сайтів. Одним з найбільш поширених прикладів мультисайтинга може бути використання загальної бази даних користувачів на декількох сайтах. Мультисайтинг це спосіб використовувати енергію відвідувачів до кінця. На звичайних сайтах відвідувачі йдуть на інші сайти Інтернету. При мультисайтингу у відвідувача є можливість продовжити серфінг на одному з сайтів зв'язки. З огляду на, що всі сайти зроблені на одному скрипті і влаштовані приблизно однаково, відвідувач відчуває більше комфорту при серфінгу/шопінгу і йому легше зробити покупку або іншу оплачуваний дію []. Розрізняють 2 види мультисайтингу: Мультисайтинг із загальним скриптом. Багато незалежних сайтів використовують один скрипт. Мультисайтинг з загальними таблицями. Багато сайтів частково використовують однакові таблиці в базі, наприклад, дані про користувачів. Варіанти загального скрипту: Якщо сайтів не багато (2-5), то можна завести окрему папку на сервері, куди завантажувати еталонну версію скрипта і всіх модулів. Мультисайтинг із загальним двигуном на бекап модулі (програма, яка робить [backup](#), резервну копію). При оновленні скрипту переносити вручну або за допомогою модуля еталонну версію по локальних папках. Звідки вони будуть заливатися на сайт при черговому оновленні сайту. Мультисайтинг із загальним двигуном на [FTP](#)-синхронізаторі. Якщо ми не збираємося налагоджувати локально сайти з використанням останньої версії скрипту, то можна використовувати мультисайтинг із загальним двигуном ще простіше. Береться [FTP](#)-синхронізатори з його допомогою заливається вміст еталонної папки безпосередньо на сайти. Мультисайтинг із загальним скриптом засобами сервера. Якщо частина сайтів розташована на одному сервері, то з'являється можливість перенести папку з еталонним двигуном на сервер. Фізично папка з скриптом буде одна для всіх сайтів. Сервер сам буде

Цитування: 0%

id: 28

«розмножувати»

цю папку для всіх сайтів. Варіанти з загальними таблицями []. Якщо кожен сайт зберігає таблиці у своїй незалежній базі, то сайти повністю незалежні (з точки зору відвідувачів). Сайти об'єднують свою базу користувачів, але зберігають незалежність в іншому. Тут таблиці, що відповідають за користувачів, у сайтів загальні, а все інше різне. Сайти об'єднують користувачів і частина вмісту. Наприклад, у різних сайтів загальні користувачі і загальний форум. Статті різні на кожному сайті. Пошуковики не люблять, коли один і той же вміст з'являється в різних місцях. Їм незрозуміло, що є першоджерелом. В результаті буде показано вміст тільки на одному сайті. Причому вибір може бути зроблений випадково. Так що відвідувач запросто може потрапити на найслабший сайт з пов'язаних в мультисайтингу. У найгіршому випадку пошуковики можуть покарати інші сайти за спробу обману пошукових систем. Відвідувачам теж незручно мати справу з частковим зеркалюванням сайту. Посилання на дзеркальний вміст показуються в його браузері як ще не переглянутих. Сайти об'єднуються повністю. Виходять

Цитування: 0%

id: 29

«дзеркала».

Раніше сайти дзеркала були поширені як спосіб боротьби з поганим хостингом. Якщо не було видно сайту на одному місці, то відвідувач міг знайти його на іншому. Зараз дзеркала вижили тільки для файлових архівів, в основному на безкоштовних сховищах. Комерційний сайт не повинен віддзеркалюватися. Пошуковики можуть показувати одні посилання на такий сайт і [www](#), а інші без. Відвідувач зайдет по посиланню з [www](#), перегляне сайт і побачить на форумі посилання без [www](#). Її дав той, хто зайшов за адресою без [www](#). Але браузер може показати її як ще не відвідану. І відвідувачеві піде вдруге дивитися ту ж саму сторінку. Якщо проводити кордон об'єднання таблиць, то найоптимальніше об'єднати зареєстрованих користувачів, але не замахуватися на об'єднання вмісту, тобто використовувати другий варіант зазначений вище. Сайти зберігають незалежний вміст з точки зору пошукових систем і відвідувачів. Але при цьому відвідувачі вільно переходять з одного сайту на інший зі збереженням свого

Цитування: 0%

id: 30

«обличчя»

і купівельної

Цитування: 0%

id: 31

«історії»

. При мультисайтингу на загальних таблицях зареєстрований на якомусь сайті користувач автоматично отримує можливість скористатися тим же логіном і паролем при вході на інші сайти зі зв'язки. Має також сенс об'єднати і дані профілю. Якщо сайти в мультисайтинговій

зв'язці працюють незалежно, то в такому об'єднанні немає сенсу. Користувачі не знають, що десь у світі є інші сайти, на яких можна скористатися тим же логіном/паролем треба в блоці логіна на кожному сайті зі зв'язки повісити оголошення про це і дати список сайтів зі зв'язки. Якщо відвідувач вже зареєстрований на якомусь із сайтів зв'язки, то це спонукає його використовувати наявний логін/пароль. Всі інші з цікавістю можуть пройти по посиланням, щоб подивитися, що там цікавого є. Крім об'єднання призначених для користувача логінів/паролів можна об'єднати між сайтами і базу покупців. Якщо користувач зробив покупку на одному з сайтів, то на іншому сайті він може зробити покупку в спрощеному варіанті. Всі його адреси і координати підставити з анкети, яку він заповнював при своїй першій покупці на іншому сайті. Створюючи зв'язки сайтів із загальною спрямованістю можна зберегти користувачів всередині власної мережі отримуючи множинну вигоду з кожного користувача відвідування і кліка. Залучення нового клієнта в мережу буде за визначенням коштувати завжди дорожче ніж закліцувати переходи вже прийшов клієнта, варто звернути увагу на технологію мультисайтінгу як на ефективний спосіб розширювати свою присутність на ринках і завойовувати окремі ніші. Найбільше в даному контексті дана методика підходить для торгових або медійних мереж які спроможні пропонувати користувачеві нові і нові гілки переходів на інші сайти [1]. Шаблон MVC – головні положення Шаблон проектування Model-View-Controller (далі просто MVC) ліг в основу архітектурного рішення першої середовища програмування з графічним інтерфейсом користувача - Smalltalk-80. Вперше MVC описав ще в 1978 році норвежець Трюгве Рінскауг, який працював деякий час в лабораторії Xerox PARC. Реалізацію шаблону в Smalltalk-80 Стів Бурбек [2]. Шаблон проектування MVC передбачає поділ даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно. Модель (Model) надає дані предметної області виду і реагує на команди контролера, змінюючи свій стан. Вид (View) відповідає за відображення даних предметної області (моделі) користувачеві, реагуючи на зміни моделі. Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін (рис. 1.1). Рис.1.1. Схема шаблону MVC [3] Розглянемо простий приклад MVC. Модель може являти собою об'єкт, який реалізує перемикач. У найпростішому випадку дана модель характеризується станом: вимкнений або включений – і, крім того, дозволяє змінювати його – об'єкт моделі має метод зміни стану: вимкнути і включити. Вид відображає на дисплеї стан перемикача за допомогою певної текстової або графічної форми. Наприклад, уявлення може відображати текстову мітку, яка при зміні стану моделі перемикача відобразить відповідний текст:

” Цитування: 0% id: 32

«вимкнений»

або

” Цитування: 0% id: 33

«включений».

Крім відображення уявлення дозволяє користувачеві змінювати стан перемикача за допомогою графічних примітивів, наприклад, двох кнопок з написами:

” Цитування: 0% id: 34

«Включити»

і

” Цитування: 0% id: 35

«Вимкнути».

Вид вміє тільки відображати стан моделі перемикача, для зміни виду звертається до контролера. Контролер являє собою об'єкт, який в нашому випадку вміє тільки змінювати стан моделі перемикача. Запропоноване програмістами Smalltalk-80 рішення виявилось настільки ефективним, що по закінченні вже майже 30 років з моменту своєї появи шаблон проектування MVC досі є стандартом настільних і Інтернет-додатків. В цьому легко переконатися - досить розглянути, наскільки MVC представлений в популярних платформах програмування. MVC задає не тільки правила поділу додатки на окремі компоненти, скільки правила їх взаємодії. У той час як уявлення і контролер залежать від моделі, модель не залежить ні від уявлення, ні від контролера. Це ключова особливість поділу, яка дозволяє працювати з моделлю, а значить, і з бізнес-логікою програми, незалежно від візуального представлення [4]. Вид безпосередньо управляє відображенням інформації користувачеві. Сама ця інформація в термінах шаблону проектування MVC називається моделлю. Контролер співпрацює з видом, гарантуючи інтерпретацію дій користувача над даними, знову ж таки - відображенням моделі. Тому уявлення і контролер можуть бути обмежені типом даних (моделлю), з яким вони працюють. Це обмеження дозволило використовувати уявлення і контролер спільно для роботи моделлю. Навпаки, модель не повинна мати обмежень. Обмеження на тип об'єкту, який може функціонувати в якості моделі, в свою чергу обмежить рамки використання тріади MVC. Необхідно, щоб будь-який об'єкт міг бути моделлю. Наприклад, число з плаваючою точкою може виступити моделлю для відображення температури. З іншого боку, модель може бути обмежена типом даних, який вона представляє, і який успадковується від суперкласу. В описі оригінальної реалізації MVC в Smalltalk згадується про пасивної і активної моделі. Пасивна модель не знала про існування уявлення, контролера, і навіть про свою участь в MVC-тріаді. Контролер відстежує зміни моделі і оповіщає уявлення. При цьому або контролер передає поданням інформацію про зміни, або уявлення самостійно вибирає дані з моделі. Більш витонченим рішенням є активна модель. Активність моделі проявляється в її праві самостійно оповістити уявлення про зміну свого стану. Щоб не порушити основна вимога MVC про незалежність моделі від уявлення і контролера, механізм оповіщення реалізується на основі шаблону Observer. У завдання контролера входить перетворення дій користувача в виконання певної операції над моделлю. Найбільш підходящим рішенням є реалізація шаблону проектування Command. Шаблон проектування Command (Команда) (він же Action

(Дія) або [Transaction](#) (Транзакція)) використовується, якщо необхідно надіслати об'єкту запит, не знаючи про те, виконання якої операції запрошено, і хто буде її одержувачем. Рішення полягає в тому, що запит інкапсулює об'єкт Команда, який задає інтерфейс виконання операції, визначаючи тим самим зв'язок між одержувачем і ініційованим дією.

 Обнаружен Плагиат: **0,03%**

id: **36**

<https://reporter.zn.ua/vyk-zrozumiti-prosto-chi-skladno-rech>
Ініціатор створює об'єкт Конкретна Команда і відправляє його в запиті на виконання, клієнт приймає запит, встановлює одержувача запиту і перетворює об'єкт Конкретна Команда в набір дій над одержувачем

Шаблон проектування [Command](#) розриває зв'язок між об'єктом, який ініціює операції, і об'єктом, що має інформацію про те, як операції виконувати. Крім того, створюється об'єкт, який можна розширювати через наслідування. Висновки до розділу 1 У першому розділі розглянуто найбільш широко використовувані прийоми для швидкої розробки веб-сайту, при ефективній забезпеченості зручності впровадження зміни вмісту в дані веб-системи, багатомовність, оптимізованість для пошукових систем. Проведено аналіз розробки сайтів за допомогою використання сторонніх сервісів, популярної [CMS](#); створення статичних веб-сторінок вручну та найбільш сучасних генераторів статичних веб-сторінок. Надано характеристику кожному з них. Виявлено, що при використанні сторонніх сервісів накладаються додаткові витрати на обслуговування системи, а це не гарантує повної можливості у збереженні даних при зміні сервісу. При застосуванні [CMS](#) виявлено вразливості у архітектурі, які виникають у функціоналі, через певні хибні базові концепції які заважають цілям веб-сайту. Веб-додатки написані самостійно, також значною мірою вразливі до хакерських атак і часто потребують значних витрат для своєї розробки, для вирішення певної задачі написання свого власного продукту є надмірні затрат, які сповільнюють бізнесові цілі. в тому разі якщо необхідний дешевий сайт- візитівка чи сторінка презентація, при цьому можливо користуватися статичними сторінками, які можуть допомогти заощадити на більш дешевих послугах хостингу, в тому числі і за рахунок відмови від використання баз даних. Здійснений аналіз доступності і функціональних можливостей найпопулярніших генераторів сайтів, виявив, що найбільш популярна мова їх розробки [JavaScript](#) та [PHP](#). РОЗДІЛ 2 ТЕХНОЛОГІЇ РОЗРОБКИ ЯК ОБ'ЄКТ ДОСЛІДЖЕННЯ Питанню написання сайтів присвячено велику кількість праць і практичних порад. Різні дослідники розглядають процес створення сайтів з різних боків. Р. Ніксон, висвітлює погляд на питання створення сайтів з боку людини, яка тільки почала освоювати цю професію. Він охоплює розробку з усіх сторін. Весь стек веб-технологій, починаючи зі стандарту [HTML5](#) і [CSS](#) включно з [JavaScript](#) і до [back-end](#)-у [MySQL](#) для роботи з базами даних і технології розробки динамічних сайтів [PHP](#). У кінці своєї роботи Р. Ніксон узагальнює весь матеріал і робить акцент на практичному значенні роботи. Автор допомагає створити повнофункціональний сайт, котрий працюватиме за принципом соціальної мережі. Також акцент робиться більше на скриптовій мові [PHP](#), ніж на мовах розмітки чи [front-end](#)-у – [JavaScript](#) [1]. Д. Колісниченко розглядає питання створення сайтів суто зі сторони мови розробки [PHP](#). У своїх дослідженнях [2] він концентрує увагу з практичного боку використання технології – демонструє створювання форуму, чату, стрічки новин, поштової розсилки й системи голосування на сайті. Також розглядає таке базове поняття, як двигун сайту, та можливості його створення. За його допомогою показує як розробити інтернет-магазин [3]. Про технології подання інформації, поняття універсального коду, який працюватиме однаково в усіх браузерах. розглядають. Е. Фрімен і Е. Робсон Вони показують базові конструкції технології на початку, а далі обговорюють більш складні поняття створення сайтів з боку суто мови [JavaScript](#) [4]. На прикладі інтерактивних елементів, котрі допомагають зробити сайт більш цікавим для користувача, В. Дронов досліджує створення сайтів суто зі сторони [front-end](#) технологій, а саме мов [HTML5](#), [CSS3](#) та [JavaScript](#). Розкриваються питання помірною завантаження контенту, семантичної розмітки й використання баз даних для формування вмісту веб-сторінок сайту з використанням бібліотеки [Ext Core](#). [5]. На початку свого дослідження Д. Склар розглядає операції, логіку роботи деяких службових конструкцій, оперування з об'єктами. У роботі обговорені різні методи зберігання інформації: у файлах, базах даних, куках веб-браузера користувача. Розглянуто питання роботи з [PHP](#) у командному рядку, інтернаціоналізація й локалізація сайту [6]. У роботі Д. Седерхольма і І.

 AI generated text detected: ChatGPT 5.3, : 53,37%

id: **37**

Маркоттوما описано такі питання, як використання у веб-дизайні нових властивостей [CSS3](#), гнучку роботу з кольором за допомогою [RGBA](#), роботу з плавучими елементами, використання гумової верстки й гнучких елементів дизайну, мистецтво праці з типографікою, фоновими елементами, а також масу інших аспектів з удосконалення веб-дизайну за допомогою технології [CSS](#) [7]. Особливу увагу приділено специфіці вживання [JavaScript](#) для зв'язку із сервером та написання крос-домених запитів при створенні веб-сайтів та веб- додатків реального часу у дослідженнях А.

Маккоу [8]. Дослідження створення сайтів повною мірою зі сторони сервера О. Мазуркевич і Д. Єловий вибудували на основі мови програмної розробки [PHP](#), зробили опис усіх елементів і розглянули особливості синтаксису мови створення серверних сценаріїв. Вони продемонстрували процес встановлення [PHP](#) і серверу [Apache](#). Праця розроблена таким чином, що її можна використовувати і як підручник, і як довідник для серверних веб-розробок [9]. Існує ще велика кількість наукових досліджень по сайтобудові та різноманітні думки дослідників програмування на можливості використання різних інструментів для виконання певних задач. Однак з понад сказаного, підсумовуючи зробимо висновки що основними мовами клієнтської частини сайту є [HTML5](#), [CSS3](#) та [JavaScript](#), серверну частину створюють за допомогою мови розробки [PHP](#), а для зберігання інформації та управління нею використовують бази даних і систему [MySQL](#) [10, 11, 12]. Розглянемо докладніше кожну технологію, які будемо використовувати при створенні [GMS](#) конструктору. Мова розмітки [web](#) – документів [HTML](#) [HTML](#) (від англ.

 Обнаружен Плагиат: **0,1%** <https://w3schoolsua.github.io/html/index.html>

id: **38**

[Hyper Text Markup Language](#)) - стандартна мова розмітки документів у Всесвітній павутині.

Більшість веб-сторінок містять опис розмітки на мові [HTML](#) (або [XHTML](#)). Мова [HTML](#) інтерпретується браузером і відображається у вигляді документа в зручній для людини формі. Мова [HTML](#) до п'ятої версії є додатком [SGML](#) (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту [ISO 8879](#). У свою чергу [XHTML](#) є більш строгим варіантом [HTML](#), він наслідує всі обмеження [XML](#) і, фактично, [XHTML](#) можна сприймати як додаток мови [XML](#) до області розмітки гіпертексту [1]. У всесвітній павутині [HTML](#)-сторінки, як правило, передаються браузером від сервера по протоколах [HTTP](#) або [HTTPS](#), у вигляді простого тексту або з використанням шифрування

[1]. Текстові документи, що містять розмітку на мові [HTML](#) (такі документи зазвичай мають розширення [.html](#)), обробляються спеціальними додатками, які відображають документ в його форматі вигляді. Такі додатки, звані

Цитування: 0%

id: 39

«браузерами»

або

Цитування: 0%

id: 40

«інтернет-оглядачами»,

зазвичай надають користувачеві зручний інтерфейс запиту веб-сторінок, їх перегляду (і виведення на інші зовнішні пристрої) і, при необхідності, відправки введених користувачем даних на сервер [1]. З появою [HTML5](#) [1, 2] вводять кілька нових елементів і атрибутів, які відображають типове використання розмітки на сучасних веб-сайтах. Деякі з них - семантичні заміни для використання універсальних блокових ([div](#)) і малих ([span](#)) елементів, наприклад, [nav](#) (блок навігації по сайту), [footer](#) (зазвичай відноситься до нижньої частини сторінки або останньому рядку [HTML](#) коду) або [audio](#) і [video](#) замість [object](#) [1]. На додаток до визначення розмітки [HTML5](#) встановлює [API](#) [1, 2], який може бути використаний з [JavaScript](#). Можливості [DOM](#) ([Document Object Model](#) -

Цитування: 0%

id: 41

«об'єктна модель документа»

- це незалежний від платформи і мови програмний інтерфейс, що дозволяє програмам і скриптам отримати доступ до вмісту [HTML](#)-, [XHTML](#)- і [XML](#)-документів) розширені і фактично використані властивості задокументовані. Також додані нові [API](#), наприклад: елемент полотна для безпосереднього методу малювання в 2D за специфікацією [Canvas 2D API Specification 1.0](#); контроль над відтворенням медіафайлів, який може використовуватися, для синхронізації субтитрів з відео; зберігання даних в браузері; [File API](#): можливість завантаження документа через вибір (тег [input type](#) =

Цитування: 0%

id: 42

"[div](#)"

) або перетягуванням ([Drag-and-drop](#)); [Drag-and-drop](#): надає набір подій для кожного елемента [DOM](#), таких як поява і знаходження в його зоні, завдяки яким розробник може інформувати користувача про необхідні дії і ідентифікатор перетягнутого файлу, що містить адресу, ім'я, тип, розмір і дату зміни; управління історією браузера; тип [MIME](#) (([Multipurpose Internet Mail Extensions](#) - багатопільові розширення інтернет-пошти) - стандарт, що описує передачу різних типів даних по електронній пошті, а також, в загальному випадку, специфікація для кодування інформації і форматування повідомлень таким чином, щоб їх можна було пересилати через Інтернет) і реєстрація обробника протоколу [1]. [XHTML5](#) - це [XML](#)-серіалізація мови [HTML5](#). Документи [XML](#) повинні бути забезпечені [XML Internet media type](#), наприклад, [application / xhtml](#) + [xml](#) або [application / xml](#) [1]. [XHTML5](#) вимагає суворого і правильно оформленого синтаксису [XML](#). Вибір між [HTML5](#) і [XHTML5](#) зводиться до вибору типу [MIME](#) / вмісту: тип медіа, який буде обраний, визначати, який тип документа повинен бути використаний [1]. Каскадні таблиці стилів - [CSS CSS](#) ([Cascading Style Sheets](#) - каскадні таблиці стилів) - формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки переважно використовується як засіб опису, оформлення зовнішнього вигляду веб-сторінок [1]. [CSS](#) використовується творцями веб-сторінок для завдання кольорів, шрифтів, розташування окремих блоків і інших аспектів представлення зовнішнього вигляду цих веб-сторінок. Основною метою розробки [CSS](#) було розділення опису логічної структури веб-сторінки (яке проводиться за допомогою [HTML](#) або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер проводиться за допомогою формальної мови [CSS](#)) [1]. Правила [CSS](#) пишуться на формальній мові [CSS](#) і розташовуються в таблицях стилів, тобто таблиці стилів містять у собі правила [CSS](#). Ці таблиці стилів можуть розташовуватися як у самому веб-документі, зовнішній вигляд якого вони описують, так і в окремих файлах, що мають формат [CSS](#). Тобто ці таблиці стилів можуть бути підключені, впроваджені в описуваний ними веб-документ чотирма різними способами) [1]. 1. Коли таблиця стилів знаходиться в окремому файлі, вона може бути підключена до веб-документу за допомогою тега [link](#), розташованого в цьому документі між тегами [head](#) і [/head](#). Усі правила цієї таблиці діють протягом усього документа. Цей спосіб підключення показаний у коді) [1]: `!doctype html html head ..... link rel="stylesheet" href="style.css" /head body ..... /body /html` 2. Коли таблиця стилів знаходиться в окремому файлі, вона може бути підключена до веб-документу за допомогою директиви [@import](#), яка розташовується в цьому документі між тегами [style](#) і [/style](#) та яка вказує у своїх дужках після слова [url](#) на адресу цієї таблиці стилів, як у коді нижче. Усі правила цієї таблиці діють протягом усього документа) [1]. `!doctype html html head ..... style media="all" @import url(style.css); /style /head /html` 3. Коли таблиця стилів описана в самому документі, вона може розташовуватися в ньому між тегами [style](#) і [/style](#). Цей спосіб показаний у коді далі. Усі правила цієї таблиці діють протягом усього документа [51]. `!doctype html html head ..... style body { color: red; } /style /head /html` 4. Коли таблиця стилів описана в самому документі, вона може розташовуватися в ньому в тілі якогось окремого тега цього документа. Усі правила цієї таблиці діють тільки на вміст цього тега. Цей спосіб продемонстрований у коді [51]: `!doctype html html head ..... head body p style="font-size: 20px; color: green; font-family: arial, helvetica, sans-serif" ..... /p /body`

/html У перших двох випадках до документа застосовані зовнішні таблиці стилів, а в інших двох – внутрішні таблиці стилів. Для додавання CSS до XML-документу останній повинен містити спеціальне посилання на таблицю стилів. Наприклад: `?xml-stylesheet type="text/css" href="style.css" [ ]`. До появи CSS оформлення веб-сторінок здійснювалося виключно засобами HTML безпосередньо всередині вмісту документа. Однак з появою CSS стало можливим принципове розділення змісту й представлення документа. За рахунок цього нововведення стало можливим легке застосування єдиного стилю оформлення для маси схожих документів, а також швидка зміна цього оформлення. Переваги CSS полягають у наступному. 1. Кілька дизайнів сторінки для різних пристроїв перегляду. Наприклад, на екрані дизайн буде розрахований на велику ширину, під час друку меню не виводитиметься, а на ПК і стильниковому телефоні меню буде йти за вмістом. 2. Зменшення часу завантаження сторінок сайту за рахунок перенесення правил представлення даних у окремий CSS-файл. У цьому випадку браузер завантажує тільки структуру документа та дані, що зберігаються на сторінці, а представлення цих даних завантажується браузером тільки один раз і може бути закешоване (збережене). 3. Простота подальшої зміни дизайну. Не потрібно правити кожен файл, а лише змінити CSS-файл. 4. Додаткові можливості оформлення. Наприклад, за допомогою CSS-верстки можна зробити блок тексту, який буде обтікати зовнішній текст, або зробити так, щоб меню було завжди видно при прокручуванні сторінки []. Водночас у CSS існують недоліки. 1. Різні відображення верстки в різних браузерах (особливо застарілих), які по-різному інтерпретують одні й ті самі правила CSS. 2. Часто зустрічається необхідність на практиці виправляти не тільки один CSS-файл, але й теги HTML, які складним способом пов'язані із селекторами CSS, що іноді зводить нанівець простоту застосування єдиних файлів стилів і значно подовжує час редагування й тестування []. Найбільш повно підтримують стандарт CSS браузери, що працюють на двигунах Gecko (Mozilla Firefox), WebKit (Safari, Arora, Google Chrome) і Presto (Opera). Колись найпоширеніший браузер Internet Explorer 6 підтримує CSS не повністю. Internet Explorer 7, який вийшов через 7 років після свого попередника, хоча й значно поліпшив рівень підтримки CSS, але містить значну кількість помилок. У Internet Explorer 8 використовується новий движок, який повністю підтримує CSS 2.1 і частково – CSS 3 []. Для перевірки підтримки браузером веб-стандартів, у тому числі й різних частин стандарту CSS, був розроблений тест Acid. Його друга версія називається Acid2, а третя, відповідно, Acid3 []. Багато веб-майстрів для кросбраузерності стилів не використовують нововведення в CSS3, замінюючи їх зображеннями. Наприклад, замість заокруглення кутів використовують фонове зображення, на якому змальований цей блок без змісту (тексту) з заокругленими кутами.

Цитування: 0,01%

id: 43

«CSS3 – розширення CSS 2.1 – додає потужну функціональність до існуючих можливостей» [, . 16]

У стандартах CSS від Консорціуму W3C використовується модель, у якій властивість width визначає ширину вмісту блоку, не включаючи до неї відступи й рамки. Ранні версії Internet Explorer 4 і 5, реалізували власну модель, у якій width визначає відстань між рамками блоку, включаючи внутрішні відступи – padding і рамки – border. Крім Internet Explorer 5 цю модель так само розуміють браузери Netscape 4 і Opera 7. Підтримка стандартної моделі W3C з'явилася в Internet Explorer тільки в шостій версії []. У розробці стандарту CSS3 для вирішення цієї проблеми введено властивість box-sizing, зі значеннями content-box для вказівки на використання стандартної моделі W3C і border-box для використання моделі Internet Explorer 5. У браузері Mozilla Firefox за підтримки цієї властивості під власною

Цитування: 0%

id: 44

«робочою»

назвою -moz-box-sizing ввели ще одне значення – padding-box, таким чином створивши третю блокову модель, у якій width – це розмір вмісту та відступів блоку, які не включають рамки. Відмінності в реалізації CSS різними браузерами змушують веб-розробників шукати рішення, як примусити всі браузери відображати сторінку однаково. CSS-фільтри дозволяють вибірково застосовувати стилі до різних елементів. Такий поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Крім того, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друковане подання, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Брайля. Як відомо, HTML-документи будуються на підставі ієрархії елементів, яка може бути наочно представлена в деревовидній формі. Елементи HTML один для одного можуть бути батьківськими, дочірніми, елементами-предками, елементами-нащадками, сестринськими. Елемент є батьком іншого елемента, якщо в ієрархічній структурі документа він знаходиться відразу, безпосередньо над цим елементом. Елемент є предком іншого елемента, якщо в ієрархічній структурі документа він знаходиться десь вище цього елемента. В CSS можуть задаватися за допомогою селекторів не лише поодинокі елементи, але і елементи, які є нащадками, дочірніми або сестринськими елементами інших елементів. Динамічна мова JavaScript JavaScript (JS) – прототипна-орієнтована скриптова мова програмування. Є діалектом мови ECMAScript. JS зазвичай використовується як вбудований мова для програмного доступу до об'єктів додатків []. Найбільш широко застосування знаходить в браузерах як мова сценаріїв для додавання інтерактивності веб-сторінок. Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу. На JS вплинули багато мов, при розробці була мета зробити мову схожою на Java, але при цьому легким для використання не програмістів. Мовою JS не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних в Інтернет []. Мова JS використовується для: написання сценаріїв веб-сторінок для надання їм інтерактивності; створення односторінкових веб-застосовань (ReactJS, AngularJS, Vue.js); програмування на стороні сервера (Node.js); стаціонарних застосовань (Electron, NW.js); мобільних застосовань (React Native, Cordova); сценаріїв у

прикладному ПЗ (наприклад, у програмах зі складу [Adobe Creative Suite](#) чи [Apache JMeter](#)); всередині [PDF](#)-документів та ін. [1]. У листопаді 1996 року [Netscape](#) заявила, що відправила [JS](#) в організацію [Ecma International](#) для розгляду мови як промислового стандарту. У результаті подальшої роботи з'явилась стандартизована мова з назвою [ECMAScript](#). У червні 1997 року, [Ecma International](#) опублікувала першу редакцію специфікації [ECMA-262](#). Рік по тому, у червні 1998 року, щоб адаптувати специфікацію до стандарту [ISO/IEC-16262](#), були внесені деякі зміни та випущена друга редакція. Третя редакція побачила світ у грудні 1999 року [1]. Четверта версія стандарту [ECMAScript](#) так і не була закінчена, тому четверта редакція не вийшла. Натомість, п'ята редакція з'явилась в грудні 2009 року. У червні 2015 року вийшла шоста версія, починаючи з якої комітет [ECMAScript](#) прийняв рішення перейти на щорічні оновлення, і нова версія отримала назву [ES2015](#). Вона отримала цілу низку нововведень, серед яких: об'єкт [Promise](#) для зручного асинхронного виконання коду, деструктуруюче присвоєння, стрілочні функції, функції-генератори, шаблонні рядки, оператори оголошення змінних [let](#) та [const](#) тощо [1]. Версія [ES2016](#) вийшла у червні 2016 року, серед нововведень оператор піднесення до ступеня [**](#) та метод [Array.prototype.includes](#), який перевіряє, чи міститься переданий аргумент у масиві. Версія [ES2017](#), що вийшла у червні 2017 року, й до сьогодні є актуальною версією стандарту, яка додала можливість використання асинхронних функцій, ком, які висять у параметрах функцій, об'єкта [Atomics](#), декількох нових методів для роботи з рядками [1]. [JS](#) наразі є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професійних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-аматорів. Поява [AJAX](#) змінила ситуацію та повернула увагу професійної спільноти до мови, а подальші модифікації мови за стандартами [ES2015](#) та [ES2017](#) внесли багато корисних можливостей, яких не вистачало для ефективного програмування.

Цитування: 0,02%

id: 45

„Уперше про [AJAX](#) заговорили після появи в лютому 2005 року статті Джесі Джеймса Гарретта ([Jesse James Garrett](#)) «Новий підхід до [Web](#)-додатків» [1, с. 44]

. У результаті були розроблені та покращені багато практик використання [JS](#) (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання [JS](#) поза браузером.

Цитування: 0,03%

id: 46

«Суть технології [AJAX](#), як це і впливає з її назви, полягає в створенні запитів сервера, які виконуються в асинхронному режимі. Це означає, що серверна частина веб-додатків, що використовує [AJAX](#), працює незалежно від клієнтської частини» [1, с. 429]

Цитування: 0,04%

id: 47

«Оскільки [JavaScript](#) є мовою, що інтерпретується, дуже часто він позиціонується як мова сценаріїв, а не як мова програмування, при цьому мається на увазі, що мови сценаріїв простіше й більшою мірою орієнтовані не на програмістів, а на звичайних користувачів» [1, с. 21]

. Назва [AJAX](#) – це акронім, що розкривається як [Asynchronous JS and XML](#) і що означає асинхронний [JS](#) і [XML](#). «Якщо ця назва, на ваш погляд, мало про що говорить, ми погодимося з вами. Простіше кажучи, можна вважати, що [AJAX](#) – це «[JavaScript](#) з розширеними правами», оскільки за своєю суттю ця технологія є сценаріями на мові [JavaScript](#), які в міру необхідності у фоновому режимі виконують запити до сервера і отримують додаткові дані, оновлюючи окремі частини сторінки й тим самим виключаючи необхідність повторного її завантаження цілком» [1, с. 24]. [JS](#) має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів у ній відрізняється від традиційних мов об'єктно-орієнтованого програмування (ООП). Крім того, [JS](#) має ряд властивостей, притаманних функціональним мовам, – функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання ([closures](#)), – що додає мові додаткову гнучкість. [JS](#) має [C](#)-подібний синтаксис, але в порівнянні з мовою [C](#) має такі докорінні відмінності: об'єкти з можливістю інтроспекції та динамічної зміни типу через механізм прототипів; функції як об'єкти першого класу; обробка винятків; автоматичне наведення типів; автоматичне прибирання сміття; анонімні функції [1]. [JS](#) містить декілька вбудованих об'єктів: [Global](#), [Object](#), [Error](#), [Function](#), [Array](#), [String](#), [Boolean](#), [Number](#), [Math](#), [Date](#), [RegExp](#). Крім того, [JS](#) містить набір вбудованих операцій, які не обов'язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм [1]. Синтаксис [JS](#) в основному відповідає синтаксису мови [Java](#) (тобто, зрештою, успадкований від [C](#)), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. [jQuery](#) [jQuery](#) – бібліотека [JS](#), що фокусується на взаємодії [JS](#) і [HTML](#). Бібліотека [jQuery](#) допомагає легко отримувати доступ до будь-якого елемента [DOM](#), звертатися до атрибутів і вмісту елементів [DOM](#), маніпулювати ними. [jQuery](#) надає зручний [API](#) для роботи з [AJAX](#) [1]. Точно так же, як [CSS](#) відокремлює візуалізацію від структури [HTML](#), [jQuery](#) відокремлює поведінку від структури [HTML](#). Наприклад, замість прямої вказівки на обробник події натискання кнопки управління передається [jQuery](#), яка ідентифікує кнопки і потім перетворює його в обробник події кліка. Такий поділ поведінки і структури також називається принципом ненав'язливого [JS](#) [1]. Бібліотека [jQuery](#) містить функціональність, корисну для максимально широкого кола завдань. Проте, розробниками бібліотеки не ставилося завдання суміщення в [jQuery](#) функцій, які підійшли б усюди, оскільки це призвело б до великого коду, велика частина якого не затребувана. Тому була реалізована архітектура компактного універсального ядра бібліотеки і плагінів. Це дозволяє зібрати для ресурсу саме ту [JS](#)-функціональність, яка на ньому була б затребувана [1]. [jQuery](#) надає допоміжний метод [post](#), призначений для відправки даних сервера. Метод [post](#) отримує кілька параметрів (у числі яких [URL](#)-адреса, за якою відправляється інформація), передану інформацію та функцію-обробник, яка виконується після завершення відправки [POST](#) [1, с. 363]. Синтаксис [jQuery](#) розроблено, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів [DOM](#), створенню анімації, обробки подій і розробки [AJAX](#)-застосувань. [jQuery](#) також надає можливості для

розробників у створенні плагінів у верхній частині бібліотеки [JavaScript](#). Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня [1]. Плюси [jQuery](#): - малий розмір дистрибутива; - низький поріг входження, вичерпна документація в інтернеті; - лаконічний синтаксис; - легко розширюваний синтаксис. Мінуси [jQuery](#): - уповільнена робота програм; - проблеми сумісності з браузером. Скриптова мова [web](#)-сценаріїв PHP [PHP](#) - технологія розробки сценаріїв, що включає: [CGI](#)-інтерфейс, інтерпретатор мови; набір функцій для доступу до баз даних і різних об'єктів [WWW](#). [PHP](#) був задуманий в кінці 1994 року Расмусом Лерддорфом ([Rasmus Lerdorf](#)). Ранні не випущені версії використовувалися на його домашній сторінці для того, щоб стежити за тим, хто переглядав його інтерактивне резюме. Перша використовувана версія стала доступна на початку 1995 року і була відома як [Personal Home Page Tools](#). Вона складалася з дуже спрощеного движка синтаксичного аналізатора, який розумів тільки кілька спеціальних макрокоманд і ряд утиліт, які потім були використані на домашніх сторінках [1]. В даний час прижилася інша розшифровка цієї аббревіатури: [Hypertext Preprocessor](#). Досить важко дати будь-яку статистику розвитку даної мови, але 26 листопада 2020-го була випущена [PHP](#) версії 8.0 [1]. Головними нововведеннями стали: підтримка [union](#)-типів [1], [JIT](#)-компіляція і атрибути (також відомі як анотації) [1]. Менеджер проекту [PHP](#) в [Microsoft](#) Дейл Хирт ([Dale Hirt](#)), в розсилці [php.internals](#) випустив повідомлення про те, що після випуску версії [PHP](#) 8.0 [Microsoft](#) припинить підтримку розробки цієї мови програмування для [Windows](#). Фахівці [Microsoft](#) займалися компіляцією бінарних версій інтерпретатора для ОС [Windows](#) і тестуванням їх безпеки. У співтоваристві розробників [PHP](#) повідомили, що вживе всіх необхідних заходів, щоб знайти найближчим часом альтернативний варіант для організації підтримки [PHP](#) 8.0 і вище для [Windows](#), своїми силами. В основі своїй синтаксис [PHP](#) дуже схожий на синтаксис [C](#), [Java](#) і [Perl](#), проте простіше цих мов. Має відкритий вихідний код. [PHP](#), як і всі процедурні мови, можна розділити на, власне, мову-інтерпретатор і бібліотеку функцій. Існує велика кількість інструментальних засобів для [PHP](#): інтерфейси до всіх популярних СУБД, до поштових протоколів, до пам'яті, що, до графічних файлів, до архівів і безліч інших інструментів. Найчастіше [PHP](#)-сценарії вбудовані в [HTML](#)-розмітку всередині спеціальних тегів ? [Php](#)? . Під час запити документа, що має [PHP](#)-сценарії, на сервері відбувається виконання коду, а користувач отримує в браузері

Цитування: 0%

id: 48

«чистий»

[HTML](#). Таким чином, [PHP](#)-сценарії вирішують все ті завдання, які характерні для типових [CGI](#)-додатків. Крім обмежувачів ? [Php](#)? , допускається використання скороченого варіанту ? . Крім того, до версії 7.0 допускалося використання обмежувачів мови програмування [ASP](#) %% і [script language](#) =

Цитування: 0%

id: 49

"[php](#)"

/ [script](#) . Робота скорочених конструкцій визначається в конфігураційному файлі [php.ini](#). Імена змінних починаються з символу \$, тип змінної оголошувати не потрібно. Імена змінних і констант чутливі до регістру символів. Імена класів, методів класів і функцій до регістру символів не чутливі. Змінні обробляються в рядках, ув'язнених в подвійні лапки, і [heredoc](#)-рядках (рядках, створених за допомогою оператора [`](#)). Змінні в рядках, ув'язнених в одинарні лапки, не обробляються. [PHP](#) розглядає перехід на новий рядок як пробіл, так само як [HTML](#) і інші мови з вільним форматом. Інструкції поділяються за допомогою крапки з комою (;), за винятком деяких випадків, після оголошення конструкції [if](#) / [else](#) і циклів. Починаючи з п'ятої версії [PHP](#) має повну підтримку ООП. Робота з класами була оптимізована і тепер такий код працює досить швидко. Клас у [PHP](#) оголошується за допомогою ключового слова [class](#). Методи і властивості класу можуть бути загальнодоступними ([public](#), за замовчуванням), захищеними ([protected](#)) і приватними ([private](#)). [PHP](#) підтримує всі три основні механізми ООП - інкапсуляцію, поліморфізм підтипів і спадкування (батьківський клас вказується за допомогою ключового слова [extends](#) після імені класу). Підтримуються інтерфейси (ставляться у відповідність з допомогою [implements](#)). Дозволяється оголошення фінальних, абстрактних методів і класів. Множинне спадкування класів не підтримується, проте клас може реалізовувати декілька інтерфейсів. Для звернення до методів батьківського класу використовується ключове слово [parent](#). Починаючи з версії 5.4.0 множинне спадкування може бути реалізовано за допомогою механізму особливостей ([Trait](#)). Особливості схожі на домішки ([Mixins](#)), за винятком того що для них не можна безпосередньо створити екземпляр. Повторне використання коду укладено у використанні коду особливості в декількох класах. Дозпускається використовувати в одному класі кілька особливостей. Механізм особливостей має засіб вирішення конфліктів імен. При запуску програми код особливості буде

Цитування: 0%

id: 50

«вкомпільован»

у код, який містить його клас[1]. [PHP](#) - це кроссплатформена технологія. Дистрибутив [PHP](#) доступний для більшості операційних систем, включаючи [Linux](#), багато модифікації [Unix](#) (наприклад, [HP-UX](#), [Solaris](#) і [OpenBSD](#)), [Microsoft Windows](#), [Mac OS X](#), [RISC OS](#), і багатьох інших. [PHP](#) підтримує більшість найбільш популярних [web](#)-серверів: [Apache](#), [Microsoft Internet Information Server](#), [Microsoft Personal Web Server](#), [Netscape](#), [iPlanet](#), [Oreilly Website Pro](#), [Caudium](#), [Xitami](#), [OmniHTTPd](#) і інші. Для більшості серверів [PHP](#) поставляється в 2-х варіантах - як модуль і в якості [CGI](#)-інтерпретатора. Крім цього, програмує на [PHP](#), розробник може віддавати перевагу як процедурного, так і об'єктно орієнтованого програмування [1]. [PHP](#) здатний генерувати не тільки [HTML](#)-документи, а й зображення різних форматів, файли [PDF](#) і [Flash](#). Він здатний формувати дані в будь-якому текстовому форматі, включаючи [XHTML](#) і [XML](#). [PHP](#) підтримує роботу з [ODBC](#) і велика кількість баз даних: [MySQL](#), [PostgreSQL](#), [Adabas D](#), [InterBase](#), [dBase](#), [FrontBase](#), [SQLite](#), [Empress](#), [mSQL](#), [Solid](#), [FilePro](#), [Direct MSSQL](#), [Sybase](#), [Hyperwave](#), [Velocis](#), [IBM DB2](#), [ODBC](#), [Unix dbm](#) , [Informix](#), [Oracle](#), [DBX](#), [Ingres](#), [Ovrimos](#) [1]. Можливе створення [PHP](#)-сценаріїв для роботи з

протоколами [LDAP](#), [IMAP](#), [SNMP](#), [NNTP](#), [POP3](#), [HTTP](#), [COM](#) (на платформах [Windows](#)), [WDDX](#) і багатьма іншими []. Особливості застосування серверів БД у [web](#)-технологіях Термін

Цитування: 0%

id: 51

«сервер баз даних»

зазвичай використовують для позначення всієї СУБД, заснованої на архітектурі

Цитування: 0%

id: 52

«клієнт-сервер».

Сервер БД обслуговує базу даних і відповідає за цілісність і збереження даних, а також забезпечує операції введення-виведення при доступі клієнта до інформації. Більшість СУБД використовують мову [SQL \(Structured Query Language\)](#) - мова структурованих запитів), так як він зручний для опису логічних підмножин БД []. Призначення [SQL \[\]](#): - створення БД і таблиці з повним описом їх структури; - виконання основних операцій маніпулювання даними (вставка, модифікація і видалення даних з таблиць); - виконання простих і складних запитів. Одна з ключових особливостей мови [SQL](#) полягає в тому, що з її допомогою формуються запити, що описують, яку інформацію з бази даних необхідно отримати, а шляхи вирішення цього завдання програма визначає сама. Кожен сервер БД може працювати на певних типах комп'ютерів і мереж. Операційними системами серверів можуть бути [MSDOS](#), [OS / 2](#), [Xenix](#), [Unix](#), [Dec VMS](#). Робочі станції користувачів зазвичай працюють під управлінням [MSDOS](#), [OS / 2](#), [Microsoft Windows](#), [Xenix](#), [Unix](#). Існують можливості змішаного використання різних ОС []. Велика частина [SQL](#)-серверів може зберігати опис БД в системному каталозі, який зазвичай доступний користувачам. Для звернення до цього каталогу використовуються [SQL](#)-запити. Реляційні СУБД можуть використовувати інформацію, що зберігається в системному каталозі для оптимізації [SQL](#)-запитів. Розмір однієї бази даних на цих серверах може сягати мільйона терабайт []. Найбільш поширеними в даний час серверами є [SQL SERVER \(Microsoft\)](#), [MySQL](#), [SQL BASE SERVER](#), [Oracle SERVER \(Sun\)](#), [IBM DB2](#), [Informix](#). [MySQL MySQL](#) - вільна система керування базами даних (СКБД). [MySQL](#) є власністю компанії [Oracle Corporation](#), що отримала її разом з поглиненою [Sun Microsystems](#), що здійснює розробку і підтримку додатка []. [MySQL](#) є рішенням для малих і середніх додатків. Входить до складу серверів [WAMP](#), [AppServ](#), [LAMP](#) і в портативні збірки серверів Денвер, [XAMPP](#). Зазвичай [MySQL](#) використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутиві входить бібліотека внутрішнього сервера, що дозволяє включати [MySQL](#) в автономні програми []. Гнучкість СКБД [MySQL](#) забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу [MyISAM](#), що підтримують повнотекстовий пошук, так і таблиці [InnoDB](#), що підтримують транзакції на рівні окремих записів. Більш того, СКБД [MySQL](#) поставляється із спеціальним типом таблиць [EXAMPLE](#), що демонструє принципи створення нових типів таблиць []. Завдяки відкритій архітектурі і [GPL](#)-ліцензуванню, в СКБД [MySQL](#) постійно з'являються нові типи таблиць. [MySQL](#) портована на велику кількість платформ: [AIX](#), [BSDi](#), [FreeBSD](#), [HP-UX](#), [Linux](#), [Mac OS X](#), [NetBSD](#), [OpenBSD](#), [OS / 2](#), [Warp](#), [SGI IRIX](#), [Solaris](#), [SunOS](#), [SCO OpenServer](#), [SCO UnixWare](#), [Tru64](#), [Windows 7](#), [XP](#), [10](#). Існує також порт [MySQL](#) до [OpenVMS](#) [, 63]. Важливо відзначити, що на офіційному сайті СКБД для вільного завантаження надаються не тільки вихідні коди, а й відкомпілювалися і оптимізовані під конкретні операційні системи готові виконувати модулі. [MySQL](#) має [API](#) для мов [Delphi](#), [C](#), [C ++](#), [Ейфель](#), [Java](#), [Лисп](#), [Perl](#), [PHP](#), [PureBasic](#), [Python](#), [Ruby](#), [Smalltalk](#), [Компонентний Паскаль](#) і [Tcl](#) бібліотеки для мов платформи [.NET](#), а також забезпечує підтримку для [ODBC](#) за допомогою [ODBC](#)-драйвера [MyODBC](#) []. Висновки до розділу 2 Виконане вище дослідження показало, що в технологіях [CSS3](#), [MySQL](#), [JS](#) і [PHP](#) застосовується підхід вбудовування програмного коду в шаблони [HTML](#)-сторінок. При запиті шаблони заповнюються динамічним вмістом, зазвичай створюваним інтерпретованою мовою сценаріїв. Підхід на основі інтерпретованих сценаріїв надзвичайно зручний при розробці [web](#) - додатків. При розробці складних програмних систем варіант вбудовування коду в шаблони сторінок ускладнює взаємодію між компонентами і ускладнює реалізацію складної архітектури. Аналіз показав, що кожна з представлених технологій має переваги і недоліки, і, як наслідок, свою область застосування. Тому поставлене завдання розробити інструмент редагування, видалення та розробки шаблонів коду сайту, автоматичної кросбраузерності для властивостей [CSS](#), будемо вирішувати обраними технологіями програмування. РОЗДІЛ 3 РОЗРОБКА ДИНАМІЧНОГО ГЕНЕРАТОРА ВЕБ-САЙТІВ 3.1. Загальний план розробки динамічного генератора веб-сайтів Визначення функціоналу динамічного генератора веб-сайтів При дослідженні предметної області у першому розділі магістерської роботи дійшли до висновку що для успішної підтримки функціонування багаторівневої системи веб-сайтів з використанням дешевих серверних послуг та підвищення рівня безпеки необхідні нові інструменти. Функціональні вимоги засобу зводяться до вимог що до інструментів, які повинні автоматично додавати префікси кросбраузерності та надавати можливість відкриття, редагування і збереження файлів з розширенням [.htm](#), [.css](#), [.js](#). Також програмне застосування повинно забезпечувати роботу з шаблонами коду: створення нових, редагування існуючих та видалення старих шаблонів. Додаток працюватиме під операційними системами [WINDOWS 7](#), [XP](#) та [10](#).

AI generated text detected: ChatGPT 5.3, : 59,36%

id: 53

Приведені технології мають свої сильні та слабкі сторони, та беручи до уваги всі засоби та технології зупинитись на тому що статичні сторінки мають безумовну перевагу у швидкості та безпеці. При цьому можна розширити розуміння статичної сторінки перейшовши від простої [HTML](#) сторінки яка віддається до браузеру користувача безпосередньо такою як вона є до більш складної структури, коли сторінка все ж генерується перед поверненням до користувача та її зміст та стан може бути заданий лише за допомогою зовнішнього контролю і не може бути змінений навіть якщо у злоумисника є доступ до файлів проекту. Така псевдо статичність при якій все ж відбувається накладення схеми даних на певний шаблон, ставить собою на меті убезпечення даних клієнту від несанкціонованого змінення.

При цьому окрім змінення важливим моментом являється такий стан кінцевого клієнта, до

якого звертається користувач, що жодні данні не можуть зберігатись на ньому безпосередньо, зберігаються в зашифрованому вигляді або у вигляді надлежних **JSON** файлів відповідно до мети мережі та рівня необхідної безпеки заданого при її розгортанні. Часто можливими причинами для блокування певних сайтів є їх вміст, авторитарні режими по всьому світу блокують доступ до інформації за допомогою судових рішень, Рациональним рішенням є часткова або повна відмова від збереження даних на ресурсах які знаходяться під загрозою закриття або видалення. Окрім втрати інформації через фізичний або віртуальний доступ до серверу та блокування ресурсу, атака на інформаційну мережу може здійснювати за допомогою так званих **DDOS** атак (**Distributed Denial-of-service attack**) так званих атак на відмову в обслуговуванні. Під такою атакою розуміється напад на комп'ютерну систему з наміром зробити комп'ютерні ресурси недоступними користувачам, для яких комп'ютерна система була призначена. Одним із найпоширеніших методів нападу є насичення атакованого комп'ютера або мережевого устаткування великою кількістю зовнішніх запитів (часто безглузвих або неправильно сформульованих) таким чином атаковане устаткування не може відповісти користувачам, або відповідає настільки повільно, що стає фактично недоступним. Тож при критичній необхідності при розповсюдженні певної інформації ризики мультиваріативні і якщо гостро стоїть необхідність у миттєвій віддачі новин чи даних мережа має мати у собі

» Цитування: 0%

id: 54

«сплячі»

елементи які неактивовані в даний момент і можуть бути введені в експлуатацію при необхідності. Так вимкнені клієнти не можуть бути просто заблоковані за допомогою адміністративного чи іншого тиску через те, що, як було відмічено вище, не зберігають жодної інформації та не мають жодного аналогу стартової сторінки. Весь акцент системи лежить на розподіленості, маскуванні та на властивості працювати з видаленими даними. Розглянувши всі ввідні виникає питання чим буде відрізнятися така мережа сайтів або окремий екземпляр від банального скрипту **RSS** стрічки налаштованого на певне джерело який би просто робив окремий внутрішній запит до нього при запиті з браузера. Відмінність в тому що і сама конфігурація і програмні властивості клієнту можуть бути отримані як дані, навіть самі джерела контенту можуть бути отримані з конфігурації, клієнт може отримати свої налаштування не лише від певного серверу адміністрування але й може сам отримати у вигляді даних модулі контролю для керування іншими клієнтами або модулі та данні для того щоб клієнт сам став джерелом даних для інших членів мережі (рис.3.1). Рис.3.1. Варіанти використання кожного окремого клієнту Варто зазначити що відокремлюється роль джерела даних та серверу управління одне від одного, звичайно у класичній **REST** системі (Рис.3.2) це може бути одним і тим же ресурсом і така схема дійсно може працювати. Робота на конфігурації клієнта може бути завантажена з будь якого засобу управління, десктопний клієнт, встановлена в веб панель адміністрування чи простий запит з командної строки. Рис.3.2. Загальна схема **REST** роботи клієнта та сервера Маючи певний набір властивостей сформулюємо визначення. Під динамічним генератором веб-сайтів розуміють клієнт-серверну архітектуру в якій клієнт конфігурації може виглядати як будь яка система здатна надсилати **POST** запити та виконує роль **CMS** яка змінює стани своїх клієнтів, видаляє чи змінює вміст або функціональне призначення. На клієнтському домені встановлено, базовану на метапрограмуванні систему яка може за командою від сервера змінювати не лише вміст але й власну структуру. Задля спрощення системи та збільшення можливостей можливо щоб вміст сторінок, конфігурацій безпосередньо у серверному кешу або за допомогою гнучкої системи у форматі **JSON** файлів для розгортання на більш простих серверах. Рис.3.3. Схема взаємодії Виконуючи команди зміни контенту по регіонам чи сегментам адміністратор динамічного генератора впливає на цілу мережу ресурсів. На прикладі лендінгів можемо використовувати перелінювку та зміну **SEO** параметрів щоб швидко реагувати на зміни у кон'юнктурі ринку. В такому разі клієнти не будуть в повну міру статичними в плані програмного забезпечення яке буде на них встановлено, та динамічне керування буде більш гнучким ніж проста зміна вмісту десятків веб-сайтів через систему автоматизації роботи з **FTP**, адже в такій ситуації все ж комусь доведеться у ручному чи напівавтоматичному режимі вносити правки у всі ці сторінки. Розглянемо переваги такої моделі перед звичним мультисайтінгом. Динамічний генератор дозволить задіяти не лише зміну контенту на одному з наших клієнтів, як це пропонують деякі системи для цього пристосовані, але я саму сутність нашого віддаленого клієнту, в тому числі і його функцію, створити на ньому окремі нові веб-сервіси для провадження наступних запитів.

AI AI generated text detected: ChatGPT 5.3, : 65,3%

id: 55

За допомогою метапрограмування можна розширювати функціональні можливості керованих доменів. Вище була розглянута **push**-схема циркуляції даних яка потребує внесення змін у клієнтські домени за спеціальною командою. На противагу їй **pull**-схема влаштовує роботу таким чином коли налаштовані клієнти самі опитують сервери, на наявність змін у налаштуваннях чи контенту, за такою схемою ми можемо позбавити клієнти **POST** інтерфейсів які можуть бути не безпечними у певних ситуаціях, і організувати з них мережу «слухачів» які б змінювали свій стан періодично за внутрішнім розкладом. В сучасному світі інформація набуває особливої цінності не лише завдяки бізнес процесам, на тлі розширення глобального громадянського суспільства, захист та поширення даних мають дуже високу актуальність. Якщо ви хочете зберегти доступ до певного контенту поширюючи його по сотням доменів чи просто прагнете відокремити своїх потенціальних споживачів в новій рекламній кампанії, методика динамічної генерації веб-сайтів дозволить зробити цю роботу зручною та ефективною Постановка завдання При вирішенні поставленого завдання магістерського дослідження розроблено концепції інструментів, які ефективно вирішують проблеми проектування.

Детально розглянуті головні технології програмування та виявлено найбільш дієві їх представники. На їх основі спроектовано нові інструменти розробки веб-додатків. При дослідженні проведено аналіз питання процесу розробки сайтів, виявлено слабкі місця, що потребують удосконалення й автоматизації. Сформовано вимоги до програмного засобу та

його функціональних можливостях й завданнях які він буде вирішувати. Для програмної реалізації обрано мови програмування [JavaScript](#) та [PHP](#), а в якості скриптових мов - [CSS](#) і [HTML](#) останніх стандартів. Загальні вимоги до ПЗ, що розроблюється, складаються з вимог до операційної системи, до зручності інтерфейсу, а також вимог до функціональної складової частини. Функціональні вимоги засобу зводяться до створення інструментів, які повинні надавати можливість відкриття, редагування і збереження файлів з розширенням [*.htm](#), [*.css](#), [*.js](#). та автоматично додавати префікси кросбраузерності. Одним з найскладніших моментів у роботі верстальника сайтів є забезпечення сумісності з безліччю браузерів – програмами для перегляду веб-сторінок (так звана кросбраузерність) [_, _]. Браузери можуть інтерпретувати по-своєму одні й ті самі елементи розмітки або правила [CSS](#), у результаті чого деякі користувачі можуть побачити вміст не так, як задумував дизайнер і очікує побачити замовник.

” Цитування: **0,02%** id: **56**

«Існує більше 130 властивостей [CSS](#), кожне з яких відповідає за певний аспект відображення елемента» [_, с. 57]

. Далі готові файли передають програмісту. Програмування сайту може здійснюватися як

” Цитування: **0%** id: **57**

«з нуля»,

використовуючи фреймворк, так і на основі [CMS](#) – системи управління сайтом. Веб-розробники часто називають [CMS](#)

” Цитування: **0%** id: **58**

«движком».

Вибір мови програмування у цьому випадку – питання не принципове. Завершальним етапом розробки є тестування. Воно може включати в себе найрізноманітніші перевірки: вид сторінки зі збільшеними шрифтами, за різними розмірами вікна браузера, а також юзабіліті-тестування [_, _]. Виявлені помилки відправляються на виправлення до тих пір, поки не будуть усунені. Терміни контролює менеджер проекту. Також на цьому етапі залучають до роботи дизайнера, щоб він провів авторський нагляд. Файли сайту розміщують на сервері провайдера й виробляють потрібні налаштування. На цьому етапі сайт закритий для відвідувачів. Внутрішня [SEO](#)-оптимізація починається з визначення семантичного ядра. Тут визначаються такі ключові слова, які дозволять залучити найбільш зацікавлених відвідувачів та за якими виграти конкуренцію простіше. Потім ці слова вносяться до сайту. Тексти, посилання, інші теги адаптуються так, щоб пошукові системи могли їх успішно знаходити за ключовими словами. Зовнішня [SEO](#)-оптимізація зводиться, як правило, до побудови структури вхідних посилань. Це власне і є розкрутка. [SEO](#)-оптимізація поділяється на

” Цитування: **0%** id: **59**

«білу»

та

” Цитування: **0%** id: **60**

«чорну»

(таку, після якої сайт за два тижні потрапляє в топ, а потім у бан пошукових систем). Справжня,

” Цитування: **0%** id: **61**

«біла»

[SEO](#)-оптимізація – це трудомісткий і тривалий процес, вартість якого може в кілька разів перевищувати витрати на створення сайту. Замовник або його довірена особа переглядають готовий проект і, якщо їх усе влаштовує, підписують документи про здачу проекту. Також на цьому етапі проводиться навчання представника замовника навичкам роботи в адміністративній зоні сайту. Кросбраузерність – це властивість сайту відображатися й працювати у всіх популярних браузерах ідентично [_, с. 96]. Під ідентичністю розуміється відсутність розвалів верстки та здатність відображати матеріал з однаковим ступенем читабельності. Всі вище перелічені етапи розробки будуть спрощені при впровадженні нашого програмного продукту. 3.2. Проектування та розробка основних інструментів ПЗ В розробці використано мови програмування [JS](#) та [PHP](#), а в якості скриптових мов - [CSS](#) і [HTML](#) останніх стандартів. Застосування повинно забезпечувати роботу з шаблонами коду: створювати нові, редагувати існуючі та видаляти старі шаблони. Додаток працює під операційними системами [WINDOWS 7](#), [XP](#) та 10. Етапи розробки прототипів представляє собою створення маленьких програм, які розроблені суто для тестування функцій інструментів. Для кожного інструменту створено унікальну програму, яка ізолює його функції від інших функцій ПЗ. Розглянемо реалізацію функціональної можливості використання операції [drag and drop](#) над будь-яким об'єктом веб-системи. Код програми представлено у додатку А.

AI generated text detected: ChatGPT 5.3, : 54,53% id: **62**

Функція приймає три аргументи: елемент, для якого треба зробити операцію переміщення; контейнер, у якому знаходиться елемент для переміщення; функція, яка буде виконуватись, коли операцію [drag and drop](#) буде завершено. Перші два названі аргументи є обов'язковими, на відміну від третього. На початку роботи для елемента, який передається першим аргументом, додаються функції, що реагують на подію активація кнопкою мишки на елемент. Далі визначаються координати елемента відносно верхнього лівого кута екрана браузера.

Потім йде перевірка на наявність атрибуту [js-drag-and-drop](#). Якщо його немає, то елемент функціонує та вставляється у контейнер перед своїм оригіналом, далі йде виклик цієї самої функції. Після перевірки елементу дописується атрибут [js-drag-and-drop](#) зі значенням [true](#). Цей елемент має абсолютне позиціонування. На рисунку 3.4 показано роботу функції [drag](#)

and drop. Рис. 3.4. Виконання функціоналу **drag and drop** Для встановлення координати елементу в положення курсору миші, викликається внутрішня функція, для документа створюються події переміщення курсору та деактивація кнопки миші. Остання подія створюється і для елементу переміщення. Коли користувач деактивує кнопку миші, усі створені події анулюються та викликається функція, яка була передана третім аргументом, якщо вона була передана. 3.3. Розробка програмного коду ПЗ Директорія проекту складається з трьох основних папок та файлу головної сторінки. Папки називаються **interface**, **modules** та **projects**. У папці **interface** містяться файли коду, які відповідають за інтерфейс усієї розробки. Усі сторінки та інформація стосовно налаштування виду додатку й залежностей міститься у файлі з розширенням **.json**. Вони можуть бути кастомізовані на розсуд користувача системи. Надано перевагу файлам для вмісту інформації, щоб комплекс можна було легко переносити на різні машини та не треба було з'єднувати його з базою даних. Папка **modules** містить у собі модулі всіх файлів, які відповідають за функціональну роботу програми. Робота всіх інструментів і форм залежить від цієї папки. Остання папка з назвою **projects** включає в себе папки з проектами, які розробляються в програмному додатку. Основні файли, з якими працює програмний комплекс, мають розширення **.html**, **.css** та **.js**. Файли з розширенням **.html** призначені для вмісту розмітки коду сайту. Розмітка являє собою структуру сайту – те, як інформація подана на веб-сторінці. Розмітка сторінки складається з тегів. Кожен тег є об'єктом **DOM** та має власні атрибути. Атрибути – це властивості об'єкту. Вони можуть зберігати різноманітну інформацію, наприклад, про адресу ресурсу, на який веде певне посилання чи адресу зображення, яке потрібно завантажити. **DOM** – це абревіатура, що розшифровується як об'єктна модель документа. За допомогою моделі браузер взаємодіє зі сторінкою та надає можливість оформлювати її. Файли з розширенням **.css** містять у собі весь код, який відповідає за стилі оформлення сайту. Код оформлення записаний за спеціальними правилами. Спочатку йде селектор елементу, який необхідно оформити. Далі у фігурних дужках через двокрапку попарно написані властивості та їх значення. Властивості відповідають за те, що потрібно змінити в об'єкті на веб-сторінці, а значення показують, як саме цю властивість буде змінено. Наприклад, властивість **color** відповідає за колір тексту в елементі та може приймати будь-які значення кольору. У **CSS** можна задати колір за допомогою **rgb** системи, шістнадцяткового номеру та назви англійськими літерами. Також існує спосіб написання кольору за допомогою функції **rgba**, у яку третім аргументом передають значення каналу альфа, який відповідає за прозорість елементу. Перші три канали – це скорочення від слів **red**, **green**, **blue**. Ці канали приймають також інтенсивність кольору цифрою від 0 до 256. Система має два режими взаємодії текстовий та візуальний. В текстовому режимі (рис 3.5), система надає користувачу можливість редагувати основні файли проекту вручну (будь то файл розширення **.js**, **.html**, **.css**, тощо). Переключатись між файлами можна користуючись вкладками (рис. 3.6), які заповнюються згідно заповненню папки проекту. Рис. 3.5. Головна сторінка веб-конструктора (в текстовому режимі) Рис. 3.6. Вкладки текстового редактору Застосовуючи концепцію **WYSIWYG (What You See Is What You Get)**, візуальний режим (рис 3.7), дозволяє редагувати проєкт використовуючи заздалегідь створені шаблони елементів. Рис. 3.7. Візуальний режим редагування Основною особливістю режиму є використання вже зазначеної та описаної функції **drag and drop**. Інтерактивне меню функцій у нижній частині сайту анімоване завдяки скрипту **circleMenu.js**, код якого наведено нижче: `var htmlItems = doc.querySelectorAll('.circle-menu .cm-item'), objItems = Object.create(null); if(htmlItems.length 0) { for (var i = htmlItems.length - 1; i -1; i--) { objItems[htmlItems[i].getAttribute('data-name')] = htmlItems[i]; } objItems['menu'].onclick = function() { if(objItems['project'].className.indexOf('cm-p') === -1) { objItems['project'].className += 'cm-p'; objItems['template'].className += 'cm-t'; } else { closeMenu(); } }; objItems['project'].onclick = function() { objItems['p-new'].className += 'cm-p-n'; objItems['p-open'].className += 'cm-p-o'; }; objItems['template'].onclick = function() { objItems['t-new'].className += 'cm-t-n'; objItems['t-settings'].className += 'cm-t-s'; }; objItems['p-new'].onclick = function() { closeMenu(); modalPNew.className += 'visible-modal'; }; objItems['p-open'].onclick = function() { closeMenu(); modalPOpen.className += 'visible-modal'; }; objItems['t-new'].onclick = function() { closeMenu(); modalTNew.className += 'visible-modal'; }; objItems['t-settings'].onclick = function() { closeMenu(); modalTSettings.className += 'visible-modal'; }; } function closeMenu() { for (var i = htmlItems.length - 1; i -1; i--) { htmlItems[i].className = 'cm--item'; } У якості допоміжних форм та модальних вікон були створені наступні компоненти веб-додатку: вікно створення шаблону (рис.3.8), що дозволяє додавати до системи новий вид шаблону, вказавши назву, селектори, HTML код та стилі CSS; Рис.3.8. Вікно створення шаблону вікно редагування параметрів шаблону (рис.3.9) – візуальний редактор шаблону дозволяє редагувати поточний вид шаблону в системі; Рис.3.9. Вікно редагування параметрів шаблону модальне вікно створення проекту (рис.3.10), яке дозволяє створювати проєкти з системи (поточний вид шаблону за замовчуванням імпортується в новий проєкт); Рис.3.10. Вікно створення проекту модальне вікно відкриття проекту (рис.3.11) – відповідає за відкриття та видалення проєктів з системи. Рис.3.11. Вікно відкриття проекту Після отримання набору протестованих готових інструментів постало завдання розробки основного програмного коду засобу та об'єднання всіх програм-інструментів у один потужний комплекс, який вирішив би всі завдання, котрі було поставлено перед ним. Через пункт меню`

Цитування: 0%

id: 63

«Вендорні префікси»

викликається реалізація автоматичного додавання префіксів для кросбраузерності сайту. 3.4. Висновки до розділу 3 У результаті дослідження розроблено програмний засіб з простим інтерфейсом, який виконує можливості візуального конструювання сайту, редагування та вставки шаблонів програмного коду при створенні сайту, з автоматичною кросбраузерністю для властивостей. Розроблений програмний продукт пройшов тестування перед фінальною стадією ухвалення його коду та роботи.

AI generated text detected: ChatGPT 5.3, : 62,43%

id: 64

ВИСНОВКИ Під час магістерського дослідження проведено роботу з аналізу методів дослідження проблематики створення сайтів, розглянуто плюси та мінуси існуючих технологічних рішень, проаналізовано найпопулярніші існуючі інструменти та проблеми, які вони вирішують. На основі існуючих рішень і виявлених проблем було створено альтернативні інструменти для вирішення завдань та зібрано їх в один програмний засіб. Розглянуто найпопулярніші інструменти розробки сайту, описано основні конструктори та генератори сайтів, ретельно проаналізовано кожний з представлених зразків додатків, представлено їх переваги і недоліки та порівняльні характеристики. Оскільки питання створення сайтів є актуальною проблемою сьогодення, виявлено теоретичні аспекти дослідження, визначено функціонал динамічного генератора веб-сайтів та методи, які використовувалися в дослідженні у процесі розробки програмного застосування. Результатом магістерської роботи є розроблений програмний засіб для розробки веб-сайтів з простим та зрозумілим інтерфейсом, який виконує функції розробки шаблонів, створення сайту з них, автоматизації кросбраузерності для всіх залежних властивостей **CSS**, розробки інструментів обробки програмного коду сайту, а також найважливішої функції усієї програми, яка відповідає за операцію **drag and drop**.

Для швидкої реалізації функцій описано дерево основних папок програмного застосування з якими файлами він працює. Розроблені інструменти виконують такі функції: розробка шаблонів, створення сайту з них, автоматизація кросбраузерності для всіх залежних властивостей **CSS**, розробка інструментів обробки програмного коду сайту, автоматичне додавання префіксів при реалізації кросбраузерності та динамічної генерації сайту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ [Jamstack](https://jamstack.org), [Static Site Generators](https://jamstack.org) : офіційний каталог. URL: <https://jamstack.org>. Генератори статичних сайтів: що це та чому вони популярні. [DOU](https://dou.ua). 2021. URL: <https://dou.ua>. [Lindgaard G., Fernandes G., Dudek C., Brown J. Attention web designers: You have 50 milliseconds to make a good first impression!](https://dou.ua) *Behaviour & Information Technology*. 2006. Vol. 25, No. 2. P. 115-126. URL: <https://landfonline.com>. [Beighley L., Morrison M. Head First PHP & MySQL: A Brain-Friendly Guide](https://landfonline.com). Sebastopol : O'Reilly Media, 2009. 812 p.

Кулеба М. Основи вебпрограмування: **PHP** та **MySQL**. Київ : Дакор, 2021. 248 с. [Resig J., Ferguson R. Pro JavaScript Techniques](https://resig.j.ferguson.r). Berkeley : Apress, 2016. 408 p. Шворак А. М. Програмування ігор на **JavaScript** : навч. посіб. Луцьк : Вежа-Друк, 2022. 156 с. [Benedetti R., Cranley R. Head First jQuery: A Brain-Friendly Guide](https://benedetti.r.cranley.r). Sebastopol : O'Reilly Media, 2011. 528 p.

Використання бібліотеки **jQuery** в сучасній веброботі. Світ комп'ютерних технологій. 2020. URL: <https://it-science.com.ua>. [Beaird J., George J., Walker S. The Principles of Beautiful Web Design](https://it-science.com.ua). 4th ed. Collingwood : SitePoint, 2020. 250 p. Тенденції розвитку ринку веброботи та фрілансу в Україні. Економіка та суспільство. 2023. № 49. URL: <https://economyandsociety.in.ua>. Володина Н. Структура вебсайта. [Webmount](http://www.webmascon.com/topics/designgeneral/111/1/). URL: <http://www.webmascon.com/topics/designgeneral/111/1/>. Воронова Ю. А., Козуб Г. О. Використання методів динамічної генерації вебсайтів. *Fundamental and Applied Research in the Modern World : Abstracts of the 5th International Scientific and Practical Conference*. Boston : BoScience Publisher, 2020. P. 277-279. URL: <https://sci-conf.com.ua/iv-mezhdunarodnaya-nauchno-prakticheskaya-konferentsiya-fundamental-and-applied-research-in-16-18-december-2020-goda-boston-ssha-arhiv/>. [Boswell D., Foucher T. The Art of Readable Code](https://sci-conf.com.ua/iv-mezhdunarodnaya-nauchno-prakticheskaya-konferentsiya-fundamental-and-applied-research-in-16-18-december-2020-goda-boston-ssha-arhiv/). Sebastopol : O'Reilly Media, 2011. 208 p. [Flanagan D. JavaScript: The Definitive Guide](https://sci-conf.com.ua/iv-mezhdunarodnaya-nauchno-prakticheskaya-konferentsiya-fundamental-and-applied-research-in-16-18-december-2020-goda-boston-ssha-arhiv/). 7th ed. Sebastopol : O'Reilly Media, 2020. 704 p. Сверстюк А. Сучасний **JavaScript**: від основ до практичного використання. Львів : Видавництво Львівської політехніки, 2022. 312 с. [Wyke-Smith C. Stylin' with CSS: A Designer's Guide](https://sci-conf.com.ua/iv-mezhdunarodnaya-nauchno-prakticheskaya-konferentsiya-fundamental-and-applied-research-in-16-18-december-2020-goda-boston-ssha-arhiv/). 3rd ed. Berkeley : Peachpit Press, 2013. 336 p. Пасічник В. В. Проектування інтерфейсів користувача вебсистем. Львів : Новий Світ-2000, 2021. 260 с. [Babich N. Information Architecture for Web Design: A Complete Guide](https://babich.n). Adobe XD Ideas. 2021. URL: <https://adobe.com>. Архітектура та структура сучасного вебсайту: правила проектування. *UX/UI Ukraine*. 2023. URL: <https://uxui.com.ua>. Шерстюк О. І. Розробка динамічних вебдодатків з використанням технологій **AJAX** та **PHP**. Вісник ЖДТУ. Серія: Технічні науки. 2019. № 2 (84). С. 142-148. Бублик В. В. Сучасні стандарти вебтехнологій: каскадні таблиці стилів **CSS3**. Київ : Наукова думка, 2020. 198 с. Пасічник В. В., Жежич П. І. Гіпертекстові технології **HTML** та моделювання стилів **CSS** : навч. посіб. Львів : Новий Світ-2000, 2022. 240 с. Крєневич А. П. **Web-технології: HTML5 та CSS3** : навч. посіб. Київ : Київський університет, 2021. 255 с. Мельник А. О. Професійне програмування на **JavaScript**: архітектура та шаблони. Львів : Видавництво Львівської політехніки, 2023. 340 с. Глибовець М. М. Об'єктно-орієнтоване програмування на **PHP** : навч. посіб. Київ : НАУКМА, 2020. 215 с. Ткаченко О. М. Проектування архітектури сучасних **Web 2.0**-додатків. Інженерія програмного забезпечення. 2021. № 3. С. 45-52. Ковальчук О. В. Методи побудови адаптивних інтерфейсів вебресурсів. Комп'ютерно-інтегровані технології. 2022. Вип. 46. С. 89-95. Свиридова Т. В. Юзабіліті вебінтерфейсів: проектування зручних для користувача сайтів. Харків : ХНУРЕ, 2021. 180 с. Голуб Б. М. Використання **JavaScript**-фреймворків та **jQuery** у веброботі. Технічні науки та технології. 2020. № 4. С. 112-119. [Bhale A. 2026 Web Dev Trends That Actually Matter](https://dev.to/socyanideisgood/2026-web-dev-trends-that-actually-matter-5520?utm_source=chatgpt.com). *Dev.to*. 2026. URL: https://dev.to/socyanideisgood/2026-web-dev-trends-that-actually-matter-5520?utm_source=chatgpt.com. Козуб В. Ю., Семенов М. А., Осадчук С. І. Аналіз впливу **WebAssembly** на продуктивність складних вебзастосунків. Вісник Херсонського національного технічного університету. 2025. Т. 2, № 3(94). С. 229-234. DOI: <https://doi.org/10.35546/kntu2078-4481.2025.3.2.28> Козуб Г. О., Сурма Ю. Ю., Артеменко О. І. Роль вебкомпонентів у побудові сучасних інтерфейсів: переваги та обмеження. Вісник Херсонського національного технічного університету. 2025. Т. 2, № 2(93). С. 175-180. DOI: <https://doi.org/10.35546/kntu2078-4481.2025.2.2.21>.

Заявление об ограничении ответственности:

Этот отчет должен быть правильно истолкован и проанализирован квалифицированным специалистом, который несет ответственность за оценку!

Любая информация, представленная в этом отчете, не является окончательной и подлежит ручному просмотру и анализу. Пожалуйста, следуйте инструкциям: [Рекомендации по оценке](#)