DOI: https://doi.org/10.36910/6775-2524-0560-2020-41-21
УДК: 004.9
Богомазюк Віталій Сергійович, магістр
Луцький національний технічний університет

ДОСЛІДЖЕНЯ СПОСОБІВ ОПТИМІЗАЦІЇ ВЕБ-САЙТУ НА БАЗІ СИСТЕМИ КЕРУВАННЯ ВМІСТОМ DRUPAL 8.

Богомазюк В. С. Дослідження способів оптимізації веб-сайту на базі системи керування вмістом Drupal 8. В статті представлено результати аналізу існуючих способів та інструментів для оптимізації веб-сайту на базі системи керування вмістом Durpal 8. Під час написання даної статті використовувалися матеріали з іноземних та вітчизняних веб-ресурсів. Проаналізовано останню інформацію про оптимізацію веб-ресурсів на базі Drupal 8/9. На основі цієї інформації були проведенні дослідження по ефективності та доцільності наявних способів оптимізації. Опрацьовано та доповнено популярні стратегії для покращення швидкодії сайту. Доведено, що застосування даних рішень покращує швидкість завантаження на 30-70% в залежності від складності проекту. Досліджено доцільність використання сторонніх сервісів по кешуванню для проектів різної складності.
Ключові слова: оптимізація веб-ресурсу, системи керування вмістом, Drupal 8, швидкодія, кешування сайту, агрегація.

Богомазюк В. С. Исследование способов оптимизации сайта на базе системы управления содержанием Drupal 8. В статье представлены результаты анализа существующих способов и инструментов для оптимизации веб-сайта на базе системы управления содержимым Durpal 8. При написании данной статьи использовались материалы с иностранных и отечественных веб-ресурсов. Проанализировано последнюю информацию об оптимизации веб-ресурсов на базе Drupal 8/9. На основе этой информации были проведены исследования по эффективности и целесообразности имеющихся способов оптимизации. Обработаны и дополнены популярные стратегии для улучшения быстродействия сайта. Доказано, что применение данных решений улучшает скорость загрузки на 30-70% в зависимости от сложности проекта. Исследована целесообразность использования сторонних сервисов по кэшированию для проектов различной сложности.
Ключевые слова: оптимизация веб-ресурса, системы управления содержанием, Drupal 8, быстродействие, кэширование сайта, агрегация.

Bogomazyuk V.S. Research of ways to optimize a website based on the content management system Drupal 8. The article presents the results of the analysis of existing methods and tools for website optimization based on the content management system Durpal 8. In writing this article used materials from foreign and domestic web resources. The latest information on web resource optimization based on Drupal 8/9 is analyzed. Based on this information, studies were conducted on the effectiveness and feasibility of existing optimization methods. Developed and supplemented popular strategies to improve site performance. It is proved that the application of these solutions improves the download speed by 30-70% depending on the complexity of the project. The expediency of using third-party caching services for projects of different complexity has been studied.
Keywords: web resource optimization, content management systems, Drupal 8, performance, site caching, aggregation.

Постановка проблеми. Основною проблемою на сьогоднішній день є низька швидкодія веб-ресурсу. З розвитком технологій розміри веб-сайтів стають все більшими, наповнення та контент стають все складнішими. Швидкодія в наш час являється основним показником для індексації та просування веб-ресурсу пошуковими системами. Під час розробки або підтримки веб-сайту на базі системи керування вмістом Drupal 8 важливо приділяти увагу оптимізації та швидкодії. Для Drupal 8 існує велика кількість інструментів та сервісів, які дозволяють оптимізувати і покращити роботу сайту, але не завжди той чи інший інструмент може бути доцільний для даного випадку, і не завжди може бути доцільним використання додаткових модулів та зовнішніх ресурсів. Тому важливо розуміти що, коли і як використовувати.

Аналіз останніх досліджень та публікацій. На даний момент основний масив інформації по Drupal 8 знаходиться в інтернет ресурсах, 90% відсотків контенту розміщена на англійській мові. Перед тим як написати дану статтю було проаналізовано декілька прикладів тактик та підходів по оптимізації веб-сайтів на базі системи контролю вмісту Drupal 8 [5,6,9]. Основним джерелом інформації по Drupal є офіційний сайт [4,9], на даному ресурсі зібрана найновіша та найбільш актуальна інформація для розробників. Однак не потрібно забувати про публікації топ-контрібютерів Drupal 8/9, в статтях [7] описані нові стратегії по оптимізації ресурсу. Аналіз останніх досліджень в галузі оптимізації [2] показав, що однією з найперспективніших технологій для веб-сайтів є техноголія BigPipe.

Виклад основного матеріалу й обгрунтування отриманих результатів. Розглянемо та проаналізуємо загальні способи оптимізації веб-сайтів. Для зручності розділимо способи оптимізації на дві групи: front-end оптимізація та back-end оптимізація. Front-end оптимізація дозволяє скоротити час завантаження веб-сайту після відповіді сервера до повного завантаження сайту в браузері клієнта. Back-end оптимізація дозволяє скоротити час між запитом до сервера та відповідями. В першу чергу потрібно дослідити оптимізацію для back-end. Для Drupal 8 основним інструментом для back-end оптимізації є кешування. За кешування в Drupal 8 відповідає Cache API.

Як правило, код закінчується рендерингами (блоками, об'єктами тощо), а контролери повертають масиви або відповіді рендерингу. Тому, як правило, ви не будете взаємодіяти з API кешу безпосередньо. Замість цього ви будете використовувати кеш візуалізації та response кеш. API візуалізації використовує метадані кешування, вбудовані в масиви рендерингу, для виконання кешування. Таким чином, API Cache не повинен використовуватися для взаємодії з кешем візуалізації (не для отримання елементів кешу, ні для створення нових). Метадані кешування, що використовуються API візуалізації (див. Попередній розділ), вони досягають аж до об'єктів Response (зазвичай HtmlResponse), які реалізують CacheableResponseInterface.

В Cache API включенні кеш теги. Теги кешу надають декларативний спосіб відстеження, які елементи кешу залежать від деяких даних, керованих Drupal. Це має важливе значення для системи управління контентом, як Drupal, оскільки той самий контент можна використовувати багато в чому. Іншими словами: неможливо заздалегідь знати, де буде використовуватися певний вміст. У будь-якому місці, де використовується вміст, його можна кешувати. Це означає, що той самий вміст можна кешувати в десятках місць. Мітки кешу передаються в наборах (порядок не має значення) рядків, тому вони набираються в strings[]. Вони є наборами, оскільки один елемент кешу може залежати від багатьох тегів кешу. Згідно з умовами, вони мають форму thing:identifier - і коли немає поняття декількох екземплярів речі, це має форму речі. Єдиним правилом є те, що він не може містити пробіли. Для прикладу:

- node:5 — кеш-тег для node з id 5 (кеш оновлюється завжди при змінні цієї ноди).
- user:3 — кеш-тег для user з id 3 (кеш оновлюється завжди при змінні цього юзера).
- node_list — список кеш-тегів для node (кеш оновлюється завжди коли будь яка node оновлюється, видаляється чи створюється).
- config:system.performance — кеш-тег для конфігурації system.performance.
- library_info — кеш-теги для бібліотек.

В Drupal може виникнути необхідність скасувати кеші на основі списків, які залежать від даних об'єкта (наприклад, оновлення візуального HTML для списку, коли об'єкт більше не існує): це можна зробити за допомогою EntityTypeInterface::getListCacheTags(), після чого недійсний будь-який повернений цим методом поряд з власним тегом. Замість того, щоб кешувати відповіді в Drupal і анулювати їх з тегами кешу, можна також кешувати відповіді в зворотних проксі (Varnish, CDN ...), а потім недійсними відповіді, які вони кешують, використовуючи теги кешу, пов'язані з цими відповідями. Щоб дозволити цим зворотним проксі-серверам знати, які теги кешу пов'язані з кожним відповіддю, ви можете надсилати теги кешу разом з заголовком.

Подібно до того, як Drupal 8 може надіслати заголовок X-Drupal-Cache-Tags для налагодження, він також може надіслати заголовок Surrogate-Keys з розділеними пробілами значеннями, як очікується деякими CDN або заголовком Cache-Tag з значеннями, розділеними комами, як очікується іншими CDN. І це також може бути зворотним проксі-сервером, який ви запускаєте самостійно, а не комерційною службою CDN.

HTTP базується на тексті. Таким чином, теги кешу також базуються на тексті. Зворотні проксі можуть вільно представляти теги кешу в іншій структурі даних внутрішньо. Обмеження значення заголовка відповіді 16 Кб було вибрано на основі 2-х факторів: А), щоб гарантувати, що він працює для випадку 99%, В) що практично досяжне. Типові веб-сервери (Apache) і типові CDN (Fastly) підтримують 16 КБ значень заголовків відповідей. Це означає приблизно 1000 тегів кешу, що достатньо для випадку 99%.

Кількість тегів кешу широко варіюється залежно від сайту та конкретної відповіді. Якщо це відповідь, яка залежить від багатьох інших речей, буде багато тегів кешу. Більше 1000 тегів кешу на відповідь буде рідкісним.

Нарешті, щось більше ніж 1000 тегів кешу, ймовірно, вказує на більш глибоку проблему: що відповідь занадто складний, що його слід розділити. Ніщо не заважає вам виходити за межі цього числа в Drupal, але це може зажадати ручного доопрацювання. Що є прийнятним для таких надзвичайно складних випадків використання. Можливо, це стосується навіть менш ніж 1000 тегів кешу.

Всебічне використання тегів кешу в Drupal 8 дозволяє Drupal 8 поставлятися з його Internal Page Cache, увімкнутим за замовчуванням. Це не що інше, як вбудований зворотний проксі. Drupal 8 забезпечує модуль Internal Page Cache, який рекомендується для малих і середніх веб-сайтів. Цей основний модуль, який увімкнено за замовчуванням, кешує сторінки для анонімних користувачів. Його можна знайти за адресою: core/modules/page_cache. Ця функція покращує продуктивність, оскільки прискорює роботу сайту. Сторінки, які запитують анонімні користувачі, зберігаються при першому запиті, а потім повторному використанні; залежно від конфігурації вашого сайту покращення

продуктивності може бути значним. Веб-сайти, які обслуговують персоналізований вміст анонімним користувачам (динамічний, за сеанс, наприклад, кошик для покупок), повинні вимкнути модуль внутрішнього кешу сторінки. Цей модуль припускає, що сторінки ідентичні для всіх анонімних користувачів. Ці веб-сайти все ще можуть скористатися перевагами модуля керування динамічними сторінками, або ж можуть зробити свою персоналізацію за допомогою JavaScript + AJAX.

На сторінці Performance (admin/config/development/performance) можна налаштувати, як довго браузери та проксі можуть кешувати сторінки. Іншої конфігурації немає. Значення, встановлене для максимального часу, коли сторінка може кешуватися браузерами та проксі, буде використовуватися заголовками Cache-Control. Цей параметр ігнорується самим кешем внутрішньої сторінки, який постійно кешує сторінки до недійсності, якщо вони не несуть заголовок Expires.

Контексти кешу забезпечують декларативний спосіб створення варіацій, залежних від контексту, для того, що потрібно кешувати. Створюючи його декларативно, код, який створює кеш, стає легше читати, і ту ж логіку не потрібно повторювати в будь-якому місці, де потрібні однакові зміни контексту. Контексти кешу передаються в наборах (порядок не має значення) рядків, тому вони вводяться до рядка. Вони є наборами, оскільки один елемент кешу може залежати від багатьох контекстів кешу (залежить від нього). Як правило, контексти кешу виводяться з контексту запиту (тобто від об'єкта Request). Більшість середовищ для веб-додатків випливає з контексту запиту. Зрештою, відповіді HTTP генеруються в значній мірі в залежності від властивостей HTTP-запитів, які їх ініціювали.Але це не означає, що контекст кешу повинен виходити з запиту - вони також можуть залежати від розгорнутого коду, наприклад, контексту кешування deployment_id.

По-друге, контексти кешу мають ієрархічний характер. Найяскравіший приклад: коли змінюється щось на користувача, безглуздо також змінювати його за дозволами (тобто, набір дозволів, які користувач має), оскільки кожен користувач вже є більш деталізованим. Користувач має набір дозволів, тому кешування кожного користувача передбачає кешування за дозволами. Тепер для найбільш цікавого аспекту: якщо одна частина сторінки змінюється на одного користувача, а інша - на дозволи, то Drupal повинен бути достатньо розумним, щоб зробити комбінацію обох: тільки змінюватись на одного користувача. Саме там Drupal може використовувати ієрархічну інформацію, щоб не створювати непотрібних варіацій.

Далі розглянемо модулі для Drupal 8 які дозволяють покращити швидкодію на стороні front-end. Для початку розглянемо стратегію якої потрібно дотримуватися під час оптимізації front-end частини:

- Зменшити JavaScript, CSS and HTML.
- Включити агрегацію JavaScript and CSS.
- Налаштувати кешування браузера.
- Увімкнути gzip стискання.
- Оптимізувати зображення.
- Використовувати lazy loading для "важких" елементів.
- Налаштувати контрольні точки для завантаження зображень згідно розширення.
- Видалити CSS код який не використовується.
- Використовувати оптимальні CSS селектори.
- Завантажувати зовнішні бібліотеки та скрипти асинхроно.

Для імплементації даної стратегії потрібно використовувати вже готові рішення для Drupal 8. Це дозволить зекономити час та ресурси.

Першим кроком оптимізації буде підключення модуля BigPipe. Завантажувати модуль не потрібно так як цей модуль з версії 8.1 у вже знаходиться в ядрі, залишається тільки включити його. Цей модуль має багато вбудованих налаштувань, але помітний ефект буде видно одразу після підключення даного модуля. Головна перевага даного модуля це завантаження "важкого контенту" асинхронно, це дозволяє не блокувати завантаження основного контенту і зменшити час до появи першого контенту. Для прикладу дана технологія використовуються в Facebook.

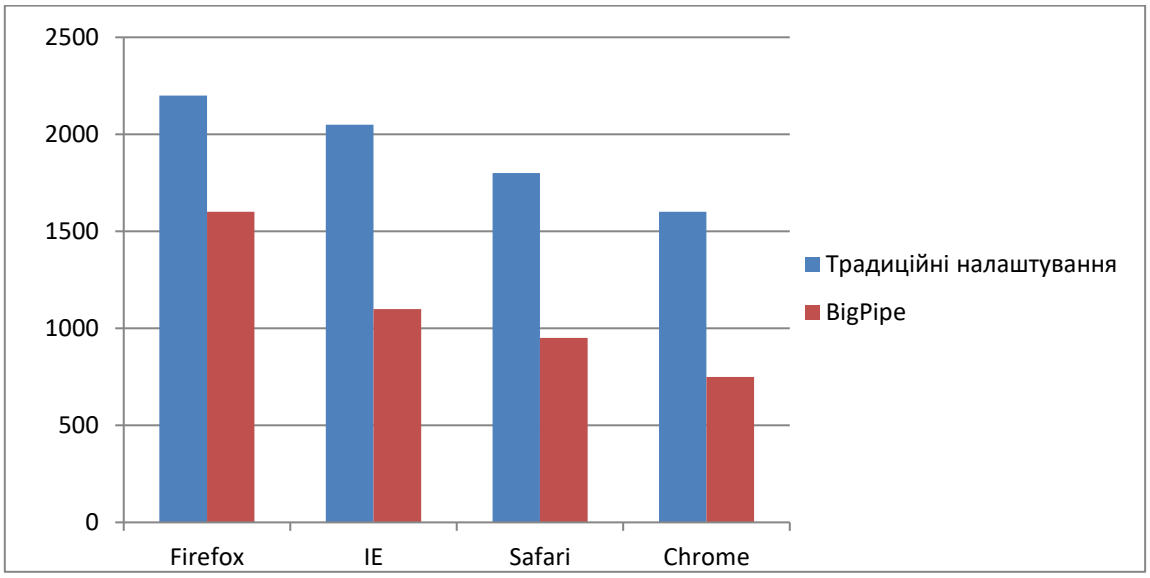

Рис 1. Порівняння скільки часу займає завантаження сторінки Facebook з та без BigPipe

Як ми бачимо з діаграми використання технології Big Pipe дозволяє зменшити завантаження веб-сайту в 2 рази. Важливий момент що в залежності від браузера відсоток на який зменшився час завантаження варіюється. Час завантаження сайту це не основний параметр за яким потрібно порівнювати швидкодію з використанням BigPipe. Декілька параметрів на які потрібно звернути увагу під час аналізу технології BigPipe:

- Time To First Byte (TTBF): Час, що пройшов між запитом HTML-сторінки та початком отримання першого байта відповіді. У цей час клієнт і браузер нічого не можуть зробити.
- Time To Interact (TTI): Повністю залежить від випадку використання, але це те, що насправді має значення.
- Page load time: Загальний час завантаження до завершення завантаження.

Після дослідження веб-сайту з використанням технології BigPipe та без використання за цими параметрами можна підвести такі підсумки:

Таблиця 1 . Порівняння основних характеристик швидкості завантаження веб-сайту

Показник	Час без використання BigPipe	З використанням BigPipe
Time To First Byte (TTBF)	4 с	1.8 с
Time To Interact (TTI)	6.3 с	3.2 с
Page load time	6.8 с	4.5 с

Як видно з досліджень технологія та модуль Big Pipe дозволяє значно покращити швидкодію веб-сайту без зайвих зусиль та фінансових витрат.

Наступним модулем який потрібно використовувати після Big Pipe це модуль Advanced CSS/JS Aggregation. Цей модуль дозволяє значною мірою оптимізувати CSS та JavaScript файли, в першу чергу за рахунок мініфікації файлів. Мініфікація це процес видалення непотрібних пробілів та зайвих розривів рядків. Також цей модуль об'єднує кілька файлів в один, це дозволяє зменшити кількість запитів на сервер та зменшити час завантаження, переносить теги підключення JavaScript файлів в кінець HTML сторінки. Цей модуль не входить в ядро Drupal 8, його потрібно встановлювати окремо.

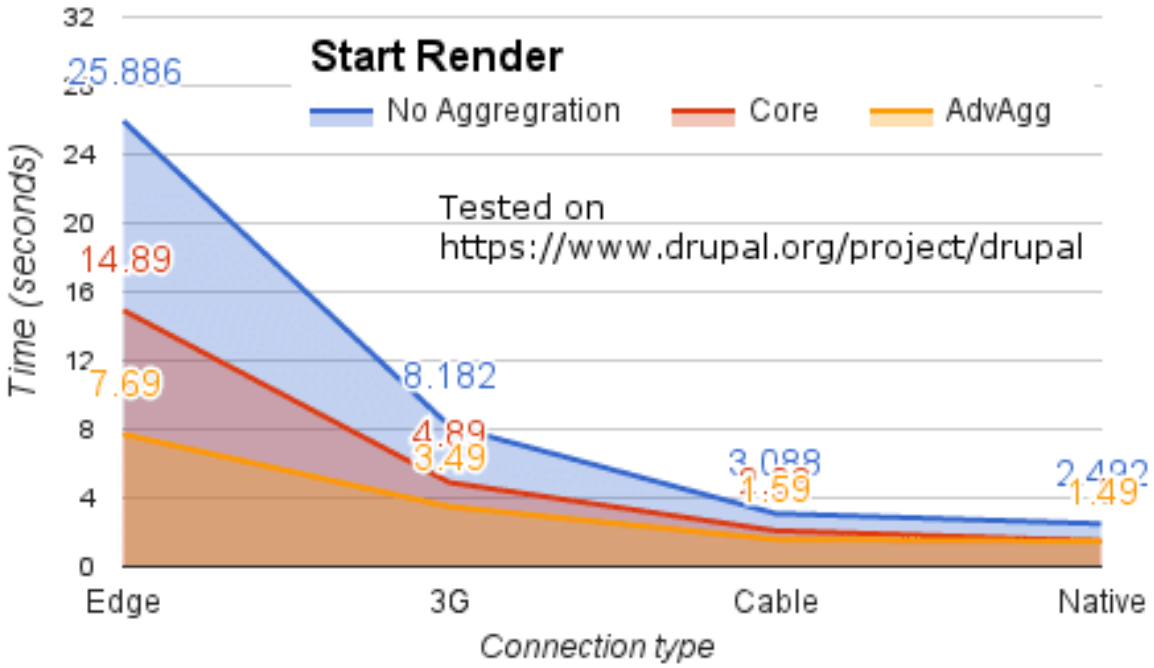

Рис 2. Порівняння швидкості завантаження сайту відносно налаштувань агрегації

На графіку ми порівнюємо сайт з трьома різними конфігураціями, перша без використання агрегації, з використанням вбудованої агрегації Drupal 8 та з використанням модуля Advanced CSS/JS Aggregation. Ці два основні модулі дозволять вагомо покращити швидкодію будь якого веб-сайту на базі систему управління вмісту Drupal 8.

Далі потрібно проаналізувати специфіку веб-сайту та контент, для прикладу якщо використовується статичний контент, можна надати перевагу модулям Dynamic Page Cache та Internal Page Cache. Модуль Internal Page Cache забезпечує внутрішнього кешування сторінок, який рекомендується для веб-сайтів малого та середнього розміру. Цей основний модуль, який увімкнено за замовчуванням, кешує сторінки для анонімних користувачів. Ця функція покращує продуктивність, оскільки пришвидшує роботу сайту. Сторінки, запитувані анонімними користувачами, зберігаються при першому запиті, а потім використовуються повторно залежно від конфігурації сайту, покращення продуктивності може бути значним. Веб-сайти, які обслуговують персоніфікований вміст для анонімних користувачів (динамічні сторінки, наприклад кошик для покупок), вимикають модуль Internal Page Cache. Цей модуль передбачає ідентичність сторінок для всіх анонімних користувачів. Веб-сайти все ще можуть скористатися перевагами модуля динамічного кешування сторінок, або ж можуть зробити свою персоналізацію за допомогою JavaScript + AJAX. Використання модулю Internal Page Cache може пришвидшити час завантаження сторінки на 70-90%, але в будь якому випадку ми втрачаємо переваги динамічного контенту.

Для веб-ресурсів з великою кількістю користувачів, з великим навантаженням на сервер потрібно використовувати стороні ресурси для оптимізації, для прикладу проаналізуємо сервіс Varnish. Varnish - це прискорювач веб-додатків, також відомий як кешований зворотний проксі-сервер HTTP. Varnish використовується на тисячах сайтів Drupal для прискорення продуктивності завантаження сторінки в 10-1000 разів, і його можна використовувати з тегами кешу, щоб полегшити інвалідацію кешу. Для веб-ресурів з невеликою кількістю користувачів недоцільно використовувати Varnish, це буде невигідно зі сторони ресурсів. У випадку коли веб-сервер отримую велику кількість запитів (1000 запитів за секунду) з однаковими параметрами є сенс розвантажити веб-сервер, саме для цього використовують Varnish.

Як відомо Drupal по замовчуванню зберігає закешовані дані в базі даних на сервері, в залежності від функціоналу 20-40% розміру бази даних може займати таблиці з зайняті кешом. У випадку потужних сайтів з великою кількістю інформації розмір бази даних може перевищувати кілька гігабайт. Проаналізуємо кілька ситуацій з різними розмірами баз даних, та яку кількість пам'яті займає кеш.

Таблиця 2. Часта таблиць кешування в базі даних

Розмір бази даних	Відсоток зайнятий кешом	Кількість пам'яті
1 GB	35%	350 MB
5 GB	30%	1.5 GB
10 GB	33%	3.3 GB
50 GB	20%	10 GB

Проаналізуємо таблицю, як видно з колонки "Кількість пам'яті", ми бачимо що велику кількість пам'яті сервера займають закешовані данні, це збільшує загальний обсяг зайнятої пам'яті на сервері, від цього збільшиться щомісячна плата за сервер. Паралельно з цим збільшується розмір бази даних, а чим більший розмір бази даних, тим більше часу займають запити до даних. Цю проблему дозволяє вирішити сервіс Redis.

Redis - це відкрите джерело (з ліцензією BSD), сховище структури даних у пам'яті, що використовується як база даних, кеш-пам'ятки та посередник повідомлень. Він підтримує такі структури даних, як рядки, хеші, списки, набори, відсортовані набори із запитами діапазонів, растрові зображення, гіперлогічні журнали, геопросторові індекси із запитами радіуса та потоками. Redis має вбудовану реплікацію, сценарії Lua, LRU, транзакції та різні рівні збереження на диску, а також забезпечує високу доступність через Redis Sentinel та автоматичне розділення за допомогою кластера Redis. В Drupal 8 є однойменний модуль Redis, який дозволяє полегшити інтеграцію з цим сервісом.

Висновки. На основних даного дослідження доходимо до висновку що оптимізація веб-сайту на базі системи керування вмісту Drupal річ специфічна, і в більшій мірі залежить від специфікації та вмісту ресурсу з яким ми працюємо. Однак, основна частина базової оптимізації для всіх веб-сайтів однакова. Перевагою Drupal 8 є готові рішення для оптимізації, контрібні модулі які покращують швидкодію, чек-лісти та багато нових модулів які без зайвих зусиль дозволяють налаштувати інтеграцію сайту з зовнішніми сервісами по оптимізації. Доведено, що застосування даних рішень покращує швидкість завантаження на 30-70% в залежності від складності проекту.

Список бібліографічного опису

1. Ayen Green. Drupal 8 Quick Start Guide. – 259 c.
2. BigPipe at Facebook: Optimizing Page Load Time– [Електронний ресурс]. - Режим доступу: https://www.infoq.com/news/2010/07/bigpipe-facebook-optimize
3. Daniel Sipos. Drupal 8 Module Development. – 231 c.
4. Drupal 8 Cache API – [Електронний ресурс]. - Режим доступу: https://www.drupal.org/docs/8/api/cache-api/cache-api
5. Drupal 8 Performance Tips and Tricks for 2020 – [Електронний ресурс]. - Режим доступу: https://thinktandem.io/blog/2020/02/04/drupal-8-performance-tips-and-tricks-for-2020
6. Elementary Tips To Optimize The Performance Of Your Drupal Website – [Електронний ресурс]. - Режим доступу: https://www.srijan.net/blog/elementary-tips-to-optimize-the-performance-of-your-drupal-website
7. Faster Web Page Loading with Facebook BigPipe – [Електронний ресурс]. - Режим доступу: https://medium.com/@arpingajjar/faster-web-page-loading-with-facebook-bigpipe-fbbc49b28959
8. Matt Glaman. Drupal 8 Development Cookbook. – 349 c.
9. Optimizing Drupal to load faster (Server, MySQL, caching, theming, HTML) – [Електронний ресурс]. - Режим доступу: https://www.drupal.org/docs/7/managing-site-performance-and-scalability/optimizing-drupal-to-load-faster-server-mysql
10. Quick Tips That Will Speed Up Your Drupal 8 Website – [Електронний ресурс]. - Режим доступу: https://www.greengeeks.com/blog/14-quick-tips-that-will-speed-up-your-drupal-8-website/
11. Simple methods to improve the page speed performance of a Drupal site – [Електронний ресурс]. - Режим доступу: https://befused.com/drupal/performance-improve
12. Website speed optimization tactics for Drupal 8 & 9 – [Електронний ресурс]. - Режим доступу: https://chromatichq.com/blog/10-website-speed-optimization-tactics-drupal-8-9