МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Методи і технології крос-платформної оптимізації сайтів

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 122 – "Комп'ютерні науки"

Науковий керівник роботи:	_____ (підпис)	Білобородова Т.О.. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Студент:	_____ (підпис)	Алдакимов А.Г. (ініціали, прізвище)
Група:		КН-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень Магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 122 – "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Алдакимову Андрію Григоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи і технології крос-платформної оптимізації сайтів

керівник проекту (роботи) к.т.н. Білобородова Тетяна Олександрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " " 2018 р. № ____
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз методів і технологій крос-платформної оптимізації веб-сайтів
2. Дослідження методів і технологій крос-платформної оптимізації
3. Розробка методу крос-платформної оптимізації веб-сайтів
4. Охорона праці в галузі

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст. викл. Критська Я.О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд та аналіз вимог до роботи.	05.09.18-19.09.18	
2	Опис сучасних методів та технологій крос-платформної оптимізації сайтів	20.09.18-04.10.18	
3	Практична реалізація розробленого методу	05.10.18-03.11.18	
4	Дослідження та аналіз отриманих результатів	04.11.18-18.11.18	
5	Розробка заходів з охороні праці.	19.11.18-03.12.18	
6	Оформлення пояснювальної записки.	04.12.18-18.12.18	
7	Підготовка та подання магістерської роботи до захисту.	19.12.18-11.01.19	

Студент _____ Алдакимов А.Г.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Білобородова Т.О.
(підпис) (прізвище та ініціали)

АННОТАЦІЯ

Алдакимов А.Г. Методи і технології крос-платформної оптимізації сайтів

Розглянуто наукові дослідження методів та технологій оптимізації сайтів. Надано теоретичні відомості про існуючі методи та технології крос-платформної оптимізації веб-сайтів. Запропоновано метод удосконалення крос-платформної оптимізації сайтів з використання визначених технологій. Викладена методика застосування запропонованого методу. Проведений експеримент з практичним застосуванням розробленого методу. Порівняльний аналіз результатів експерименту показав високі показники швидкості та продуктивності розробленого методу в порівнянні з існуючими.

Ключові слова: веб-сайт, крос-платформний, оптимізація, методи, технології, CDN, кешування, мінімізація, стиснення, швидкість, продуктивність.

АННОТАЦИЯ

Алдакимов А.Г. Методы и технологии кроссплатформенной оптимизации сайтов

Рассмотрены научные исследования методов и технологий оптимизации сайтов. Предоставлено теоретические сведения о существующих методах и технологиях кроссплатформенной оптимизации веб-сайтов. Предложен метод совершенствования кроссплатформенной оптимизации сайтов по использованию определенных технологий. Изложена методика применения предложенного метода. Проведенный эксперимент с практическим применением разработанного метода. Сравнительный анализ результатов эксперимента показал высокие показатели скорости и производительности разработанного метода по сравнению с существующими.

Ключевые слова: сайт, кроссплатформенный, оптимизация, методы, технологии, CDN, кэширование, минимизация, сжатие, скорость, производительность.

ABSTRACT

Aldakymov A. G., Methods and technologies of cross-platform optimization of sites

The scientific researches of methods and technologies of optimization of sites are considered. Theoretical information about existing methods and technologies of cross-platform optimization of websites is given. The method of improvement of cross-platform optimization of sites on the use of certain technologies is proposed. The method of application of the proposed method is outlined. An experiment was conducted with the practical application of the developed method. A comparative

analysis of the results of the experiment showed high rates of speed and productivity of the developed method compared with the existing ones.

Keywords: website, cross-platform, optimization, methods, technology, CDN, caching, minimization, compression, speed, performance.

ЗМІСТ

ВСТУП

Актуальність теми. Веб-сайти в повсякденному світі – не новинка, однак вони дуже поширенні на сьогоднішній день та зростають щоденно шаленими темпами, а саме зі швидкістю близько 200 сайтів в хвилину. При цьому з кожним роком зростає кількість користувачів. Так, наприкінці січня 2018 року глобальним медіа агентством We Are Social та розробником платформи для управління соціальними мережами HootSuite було представлено звіт, згідно з яким понад чотири мільярди людей по всьому світу використовують Інтернет [1]. Згідно цим даним, кількість користувачів Інтернету зросла до 53% населення Землі, що відрізняється на 7% в порівнянні з аналогічним періодом за 2017 рік. Також слід зазначити, що на 1995 рік доступ до Інтернету мали всього 1% населення планети. Що стосується України, то на стан 2018 року кількість користувачів складає 70% населення [2].

Однак, окрім кількості користувачів також зросла і кількість годин проведення в Інтернеті. Так, згідно останніх даних показник наскільки високий, що якщо ми складемо разом ці чотири мільярди користувачів, то ми проведемо мільярд років онлайн тільки за 2018 рік[1]. Збільшення кількості користувачів та їхньої кількості годин перебування в Інтернеті примушує розробників сайтів залучати користувачів доступними засобами. Засоби різняться своєю направленістю. Так, наприклад, деякі розробники приділяють більшу увагу на зовнішній вигляд сайту, інші ж просувають сайт завдяки SEO оптимізації.

Однак, в зв'язку з такою кількістю користувачів зростає навантаження на сайт, що впливає на його продуктивність і може привести до збоїв в роботі, що є найважливішим для функціонування сайту. Тому окрім описаних вище методів необхідно звернути увагу на продуктивність, чим найчастіше нехтують розробники, оскільки це займає багато часу. Цей недолік може дорого коштувати розробникам, оскільки якщо веб-сайт буде повільним при завантаженні та роботі, то це може привести до втрати користувачів, при цьому знизити кількість потенційних клієнтів.

Та все ж, одним з найважливіших факторів є оптимізація сайтів. Під оптимізацією розуміється процес модифікації системи для поліпшення її ефективності. Оптимізація сучасного сайту складається з багатьох різних аспектів. Що б назвати сайт оптимізованим, він повинен відповідати наступним вимогам:

- відповідатиме клієнтським запитам як можна швидше;
- бути правильно сконструйованим і простим у використанні;
- мати можливість бути використаним людьми з різними фізичними вадами;
- мати можливість бути використаним незалежно від споживчого браузера;

- бути легко знаходять сучасними пошуковими машинами.

Незважаючи на те, що велика кількість часу і грошей було присвячено збільшенню відвідуваності сайту, до сих пір мало уваги приділялося підвищенню коефіцієнта конверсії веб-сайту (тобто швидкості, з якою відвідувачі веб-сайту беруть участь в бажаної ділової активності при відвідуванні веб-сайту).

Оптимізація веб-сайту для збільшення коефіцієнтів конверсії є дуже складною справою. Одна проблема полягає в тому, що демографія відвідувачів і переваги не відомі. Інша проблема полягає в тому, що кожна сторінка веб-сайту складається з великої кількості змінних факторів, таких як вміст сторінки, візуальні елементи, посилання на заклики до дії (винесення), шрифти і макети сторінок. Поєднання цих факторів створює мільйони можливих варіацій зовнішнього вигляду певної веб-сторінки. Проте, емпіричні дані показують, що незначна зміна зовнішнього вигляду веб-сайту, наприклад, кольору кнопки «відправити», може суттєво змінити коефіцієнт конверсії веб-сторінки.

Додатковою проблемою для оптимізації веб-сайту є онлайн-маркетинг в реальному часі. Найчастіше велике маркетингове просування може бути завершено в лічені хвилини або години, не залишаючи часу для будь-яких ручних експериментів або аналізу. І, нарешті, внесення будь-яких змін в веб-сайт є дуже трудомістким процесом.

Ще одним важливим фактором для досягнення високої конкурентоспроможності є коректна робота веб-сайту на будь якій платформі, тобто крос-платформенність - здатність програмного забезпечення працювати більш ніж на одній апаратній платформі і (або) операційній системі[3]. Надання крос-платформенності сайту приводить до використання сайту без суттєвих змін коду при експлуатації веб-додатку на різних платформах.

Компанії намагаються збільшити конверсію сайту за допомогою різноманітних підходів. Деякі підходи повністю засновані на досвіді консультантів і дизайнерів, які застосовують свої багаті емпіричні знання для створення веб-сайту, який, як очікується, буде мати високий коефіцієнт конверсії. Такий підхід, однак, є суб'єктивним. Те, що є успішним для однієї компанії, може не бути ефективним для іншого. Крім того, навіть якщо нова версія більш успішна, ніж попередня, неясно, чи досяг веб-сайт максимальної конверсії.

Інші застосовують наукові методи вимірювання конверсії веб-сайту і проводять експерименти з метою знайти версію веб-сайту, яка має найвищий коефіцієнт конверсії. Найпростіший метод - це так званий метод тестування «A / B». По суті, це метод грубої сили, при якому дві (або невелика кількість) найбільш вірогідних версій веб-сайту перевіряються на предмет пошуку кращого. Зазвичай A / B-експерименти постійно проводяться в нескінченному пошуку кращого рішення. Ці експерименти трудомісткі, а їх методологія пошуку досить

неефективна. Якщо ви тестуєте кілька з мільйонів можливих варіантів однієї сторінки, легко зрозуміти, що ймовірність знаходження найкращого рішення досить низька.

Існують численні фактори, що впливають на роботу веб-сторінки через Інтернет. Хоча раніше вважалося, що час завантаження сторінки визначається насамперед сервером, проте, дослідження підтверджують, що 80-90% питань часу завантаження фактично відбуваються на передньому кінці, тобто під час рендеринга та виконання веб-сторінки браузер кінцевого користувача. Крім того, ці проблеми можна запобігти, наприклад, безліччю простої оптимізації веб-сторінки:

1. Розумна організація таблиць стилів і сценаріїв у складі сторінки
2. Мінімізація розміру або кількості запитів HTTP для об'єктів на веб-сторінці
3. Максимізація використання кешу шляхом ретельного та стратегічного визначення часу закінчення терміну дії веб-об'єкта тощо.

Такі проблеми і способи їх запобігання стають загальновідомими. Дослідники навіть сформулювали їх у набір неформальних «правил», яких потрібно дотримуватися, як в [4]. Опитування в Інтернеті також покаже, що багато компаній пропонують безкоштовні або платні послуги і інструменти по оптимізації веб-сайтів, які реалізують деякі або всі ці правила для вимірювання ефективності веб-сторінок і виявляють недоліки для поліпшення, такі як YSlow від Yahoo! та Google PageSpeed. Після того, як їх оптимізація виконується на стороні сервера і може зберігатися в CDN для ефективного розсіювання, покращений веб-контент більше не змінюється і не оптимізується динамічно на основі запитувача користувача.

Іншими словами, в даний час відсутня концепція автоматичної і динамічної оптимізації веб-сторінки на основі інформації про клієнта, яка вже відома або може бути точно виміряна тільки на пристроях, розташованих в локальній мережі клієнта або поблизу неї. Наприклад: умови локальної мережі або статистика трафіку, які відслідковуються клієнтським локальним мережевим комутатором / шлюзом. Або дані зворотного зв'язку клієнта, які застарівають після проходження всього лише кількох стрибків з локальної мережі. Або кешований вміст на прикордонному сервері можна використовувати для безпосередньої перекомпонування запитаних файлів без очікування вилучення з віддаленого центру обробки даних.

Нещодавно веб-розробники та дослідники почали зосереджуватися на оптимізації швидкості завантаження веб-сайтів. Зрозуміло, що перевагою оптимізації продуктивності веб-сайту є поліпшення досвіду кінцевих користувачів, що може призвести до збільшення трафіку на веб-сайті і, таким чином, збільшить його вартість.

Більшість дослідників, що займаються оптимізацією сайтів, працювали над SEO оптимізацією, а тих, що націлено працювали над підвищенням продуктивності і мали на меті використати повністю потенціал сайту можна виділити не так багато. Однак, серед таких

можна виділити Ron Kohavi та Alex Deng[4], М. П. Побежимова, Е. И. Казимирова, М. В. Стержанов [5], Jiang Zhu, Douglas S. Chan [6] та Francois Buchs, Zijad F Aganovic[10].

Саме обмежена кількість робіт є причиною вибору теми магістерської роботи, у якій вирішується науково-прикладне завдання удосконалення методів та технологій крос-платформної оптимізації сайтів.

Об'єкт дослідження – параметри крос-платформної оптимізації веб-сайту.

Предмет дослідження – методи та технології забезпечення високої продуктивності роботи веб-сайту.

Мета і завдання дослідження. Метою дослідження є підвищення продуктивності веб сайту, за допомогою удосконалення методів та технологій крос-платформної оптимізації веб-сайту.

Для досягнення мети дослідження необхідно вирішити наступні завдання:

– аналіз методів і технологій для оптимізації веб-сайтів;

– удосконалення обраних методів та технологій та реалізація результатів роботи у вигляді веб-сайту.

Методи дослідження. Проведені в роботі дослідження основані на технології кешування, CDN, стиснення зображень та оптимізації коду, які використовувались при розробленні таких відомих сайтів, як Microsoft, Google та багато інших.

Наукова новизна отриманих результатів:

а) Проаналізовано методи та технології оптимізації веб-сайтів.

б) Обрано та об’єднано методи та технології, які дозволяють підвищити швидкість роботи та підняти позиції сайту в пошукових системах.

Особистий внесок здобувача полягає у вдосконаленні методів та технологій оптимізації веб-сайтів, на основі проаналізованих технологій, методів та інструментальних засобів, що дозволяють вирішити поставлені задачі. Усі основні результати отримані автором особисто.

Практичне значення отриманих результатів полягає в реалізації удосконалених методів оптимізації та опису основних наукових положень дисертації.

Структура і обсяг роботи.

Магістерська робота складається зі вступу, 4 розділів, висновків на 67 сторінках, списку використаних джерел з 38 найменувань на 3 сторінках, додатків на 17 сторінках. Загальний обсяг роботи складає 96 сторінок. В магістерській роботі міститься 11 таблиць, 41 рисунків.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ І ЗАСОБІВ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕНЬ

1.1 Аналіз вимог до оптимізації

Оптимізація сайту – комплекс заходів здаля підвищення швидкості роботи сайту, використовуючи методи, в залежності від направлення. Продуктивність будь-якого веб-сайту дуже залежить від того, як він був побудований, і від якої програмної технології він використовує.

Під об’єктом дослідження мається на увазі отримання оптимізованого під різні платформи веб-сайту, який надає не тільки надає приріст в швидкості завантаження та роботи, але і при цьому забезпечує високу продуктивність. Також не менш важливо, щоб отриманий сайт займав високі позиції в пошукових системах, що виділяло б його серед конкурентів.

Веб-сайт – це сукупність веб-сторінок, які логічно взаємопов’язані [8]. Популярність веб-сайтів пояснюється тим, що сучасний світ пропонує більшість своїх товарів та послуг через мережу Інтернет, тому компанія, яка має власний веб-сайт виділяється серед своїх конкурентів. Адже саме сайт дозволяє швидко поширити інформацію для аудиторії, підвищується ефективність реклами, оскільки даний він є інструментом просування реклами. Ще однією з головних переваг є швидкий зв'язок з клієнтами, що дозволяє підвищити зацікавленість клієнтами до власника сайту.

Саме з наведених вище причин, при розробці веб-сайту необхідно звернути увагу не тільки на наповненість та рекламне просування, але й його оптимізацію [9]. Однак, слід пам’ятати, що розробка веб-додатків відбувається на системах управління контентом, таких як WordPress, Joomla та Битрикс, власними силами розробників з нуля або з використанням фреймворків, серед яких Laravel, Symphony та Angular, що приводить до необхідності підбору методів та технологій оптимізації, які б були незалежалими від платформи або способу його створення.

При виконанні даної магістерської роботи було проаналізовано існуючі методи та технології оптимізації сайтів та для удосконалення цих методів було поставлено наступні завдання:

а) Оптимізація серверної частини сайту, яка повинна підвищити його продуктивність та надійність.

б) оптимізація швидкості завантаження контенту, шляхом використання оптимізованого CSS та HTML коду.

Кожна з цих задач має свої проблеми в реалізації.

Так при оптимізації серверної частини підвищується продуктивність, однак це значення формується в результаті середнього значення масштабування, тобто розширення можливостей оброблювальних запитів в секунду, та швидкістю роботи. В результаті чого розробник вимушений поступитися в користь або масштабування, або швидкості.

Що стосується оптимізації контенту, то при ній доводиться строго притримуватися валідності коду. Ця проблема є найпоширенішої серед всіх, оскільки з нею стикаються всі розробники. На жаль, більшість її просто ігнорує, в той час як саме притримування валідності коду – основна складова крос-платформності.

1.2 Аналіз технологій і методів для вирішення задачі

На сьогоднішній день існує багато матеріалів у відкритому доступі, які дають різноманітні рекомендації щодо оптимізації, які на жаль в більшості випадків пропонують засоби для SEO-оптимізації. Однак є ряд таких, що пропонують підвищення продуктивності найрізноманітнішими методами та технологіями

Так, вирішення задачі підвищення швидкості роботи веб-сайту запропоновано авторами [5]. Дана робота пропонує використовувати методи та способи оптимізації, серед яких:

а) видалення неактуального ресурсу;

б) мінімізація HTML/CSS коду;

в) стискання зображень алгоритмами стиснення;

г) HTTP-кешування;

ґ) використання content delivery network (CDN) – технологія, що дозволяє користувачеві отримувати вміст сайту з наближених географічних місць.

Кожен з методів описаний у вигляді інструкцій з використання та має рекомендаційний характер. Особливу увагу в статті приділено кешуванню та стисненню зображень, які добре описують способи реалізації даних методів. Окрім цього, присутній аналіз CDN та описані його основні переваги. Однак, як і більшість статей по цій темі, відсутні записи про практичне застосування описаних методів, але присутні результати тестів швидкості Android-додатків.

Використовувати шаблони для досягнення крос-платформеності пропонують автори [9]. Ця стаття пропонує використовувати шаблонно-орієнтовані дизайни (ШОД), що повинні надати веб-додатку бажаного результату для різних платформ одночасно. Пояснюється це тим, що використання ЩОД приводить до вирішення проблем під час реалізації крос-платформних веб-додатків без суцільних змін коду для кожної платформи. Звернено увагу на те, що розроблені

рекомендації по створенню додатків відомими компаніями, серед яких Microsoft, IBM та Sun Microsystem, нажаль не мають рекомендації, які б не залежали від платформи.

Запропонований підхід оснований на різних архітектурних, навігаційних та інтерактивних шаблонів, що не залежать від типу платформи. В результаті дослідження авторами було запропоновано власні методи, які ґрунтувалися на шаблонах веб-дизайну, які основані на використанні різних видів шаблонів (рис. 1.1).

Рисунок 1.1 – Шаблонно-орієнтована архітектура створення веб-додатків

До шаблонного веб-дизайну належать:

а) навігаційні шаблони – навігація по сайту;

б) шаблони взаємодії – можливість взаємодіяти з елементами сайту;

в) шаблони презентації – відображення макету сторінки;

г) шаблони візуалізації – відображення графічних елементів;

ґ) шаблони сумісності – розширення систем, використання технології MVC(Model, View and Controller);

д) інформаційні шаблони – опис моделей та архітектури для реалізації базового контенту.

Основна ідея статті полягає в об’єднані описаних видів шаблонів веб-дизайну для створення адаптивного дизайну для крос-платформних веб-сторінок. Результатом статті є представлення рекомендацій використання шаблонів з прикладами та перевагами їх використання.

Автори статті [4] пропонують використовувати свої методи оптимізації. Досліджуючи існуючі методи оптимізації на відомих інтернет-ресурсах, серед яких Amazon, Facebook, Google, LinkedIn, вони виділили ряд правил, які б допомогли дослідникам оптимізувати будь-який сайт.

Всі засоби були поділені на сім емпіричних правил експериментів для веб-сайтів, які вони розробили на основі тисяч онлайн-контрольованих експериментів, і підтримувалися прикладами:

1. Незначні зміни можуть значно впливати на основні показники.

2. Зміни рідко мають великі позитивний вплив на основні показники. Перші два показують, що невеликі зміни можуть мати великий позитивний вплив, але вони рідкісні, і велика частина прогресу досягається за рахунок безлічі невеликих поліпшень з плином часу.

3. Зміни неминучі.

4. Швидкість має значення. Четверте правило - це область, якою займалися автори, а саме швидкість. Вони провели кілька експериментів, після чого було зрозуміло взаємозв'язок між продуктивністю і ключовими показниками, показуючи, що швидкість сервера має вирішальне значення; крім того, більш швидке відображення ключових частин сторінки є більш важливим, ніж інші. Це одне з найголовніших правил, оскільки продуктивність є однією з найголовніших характеристик, адже окрім зацікавленості, це також впливає на SEO-оптимізацію.

5. Зниження відмов відбувається складно, набагато простіше відбуваються переходи по клікам. Це емпіричне спостереження, яке, з часом буде уточнено, але дивно, наскільки широко воно зберігається: зміна показника відмови дійсно складно, і більшість експериментів просто змінюють кліки, тому потрібно бути обережним з локальної оптимізацією.

6. Необхідно уникати складних конструкцій. Рекомендовані більш прості конструкції і більш швидкі ітерації, що відповідає сучасним методологій гнучкої розробки програмного забезпечення.

7. Необхідно мати достатню кількість користувачів. Це забезпечує нижню межу для кількості користувачів для перекручених метрик, які є загальними в онлайн-експериментах,

тобто пропонується проводити експерименти мінімум на сайтах з мінімумом в декілька тисяч користувачів.

Дослідження проводилися на основі тисяч онлайн тестів та мають результати конкретні результати, які приведені у прикладах використання даних правил. Стаття широко описує використання кожного з правил та наводить приклади де краще застосувати те чи інше правило, однак ці правила не пропонують конкретні методи чи технології, які б можна було використати для досягнення кінцевого результату, тому стаття має лише рекомендаційний характер.

Автори статті [6] пропонують зовсім новий, не маючий аналогів, підхід, який оснований на використанні існуючих методів та архітектури Fog Computing. Fog Computing – це архітектура, яка використовує граничні пристрої для здійснення значної кількості обчислень, зберігання, локального зв'язку та маршрутизації через мережу Інтернет [7].

У порівнянні з хмарними обчисленнями, Fog Computing підкреслює близькість до цілей кінцевих користувачів і клієнтів, щільне географічне розповсюдження та об'єднання ресурсів на місцевому рівні, скорочення затримки та економію магістральної смуги пропускання для досягнення кращої якості обслуговування та крайової аналітики / потокового добування [7].

Автори застосовують прості методи в об’єднанні з Fog Computing архітектурою, що поєдує ці методи з унікальними знаннями, які доступні тільки на граничних (Fog) вузлах. Більш динамічна адаптація до умов користувача (наприклад, стан мережі та навантаження на пристрій) також може бути досягнута за допомогою спеціальних знань про межу мережі. В результаті застосування даної технології була підвищена ефективність рендеринга веб-сторінки користувача, в порівнянні з простим застосуванням цих методів.

Дане дослідження, описане в статті привело до модифікації та удосконалення методів оптимізації, шляхом об’єднання простих методів з Fog Computing архітектурою, що створило технологію оптимізації. В статті детально описано технологію оптимізації, крок за кроком. Ще однією відмінною рисою даної статті є детальний опис кожного з методів та шляхи досягнення оптимізації. Однак, як і в попередніх статтях, відсутні результати практичного застосування даної технології, тому статтю має рекомендаційний характер.

Автори патенту [10] застосували новий підхід до оптимізації веб-сайтів. Винахід включає в себе метод та пристрій, які забезпечать повністю автоматизоване, самонавчене (адаптивне) рішення для оптимізації веб-сайту на основі закритого підходу. Новий підхід не обмежений конкретної методологією оптимізації або конкретним алгоритмом оптимізації. Замість цього це платформа, яка дозволяє вам використовувати існуючі або нові винахідницькі методи або алгоритми оптимізації в циклах оптимізації веб-сайтів зі зворотним зв'язком. Метод і пристрій згідно з даним винаходом включають в себе можливість призначати змінні і вимірні цілі, можливість ініціювати процес оптимізації, а потім автоматично генерувати зміни на веб-

сайті при вимірюванні відповідей відвідувачів до тих пір, поки процес не наблизиться до кращого рішення. Автоматичні зміни веб-сайту засновані на алгоритмі оптимізації, який забезпечує конвергенцію до кращого рішення після тестування тільки обмеженого числа можливих варіантів веб-сторінки (веб-сайту). Цей метод може застосовуватися до будь-якого типу веб-сайтів, запрограмованих на будь-якому типі мови програмування, незалежно від кількості рівнів і компонентів веб-сайту, типу операційної системи і веб-сервера.

Для досягнення високої надійності і високої продуктивності даний спосіб і пристрій, що пропонується в даному винаході, забезпечують високо паралельні розподілені обчислення з унікальним способом оптимізації черг для максимального використання обчислювальних активів, а також для забезпечення того, щоб досвід кінцевого користувача не був порушений.

До переваг цього підходу можна віднести:

1. Підвищення швидкості оптимізації веб-сайту.

2. Здатність адаптуватися до динамічних змін сайту.

3. Підтримка декількох методів та алгоритмів оптимізації.

Окрім опису технології, були представлені та детально описані схеми роботи даного патенту. Патент пропонує методику для оптимізації сайту в незалежності від платформи чи мови програмування, чим має зацікавленість та велике значення для дослідників. Дане джерело надає опис універсальної платформи, яка дає можливість модифікувати та удосконалювати оптимізацію сайту, при якій не втрачається попередній прогрес розробника.

В свою чергу, автор статті [11] пропонує свої методи оптимізації сайту. Його рішенням підвищення продуктивності сайту є використання плагінів для системи управління контентом Wordpress:

1. Leverage Browser Caching – це плагін, який використовує кеш в браузерах, таких як Mozilla Firefox, для зберігання даних, зібраних на веб-даних. Недоліком даного плагіну є створення великої кількості файлів. Використання кешування може допомогти зберігати файли локально, тобто в браузері, що підвищить швидкість завантаження сайту. Хоча при першому відвідуванні цього не отримати, однак при повторному відвідуванні чи оновлення сторінки збереженні дані нададуть бажаного ефекту.

2. JS & CSS Script Optimizer – це плагін, необхідний для оптимізації файлів JavaScript та CSS, шляхом мінімізації файлів та правильному підключенню цих файлів автоматично.

3. Above The Fold – основною функцією цього плагіна є усунення рендеринга JavaScript і CSS, що перевищує складність.

4. Plugin Chace – плагін, який так як і Leverage Browser Caching займається кешуванням.

Стаття добре описує представлені рішення оптимізації та надає результати оптимізації, отримані за допомогою Google Pagespeed. Однак всі ці засоби можна застосувати лише на

сайтах, що створені на базі CMS WordPress, що не є актуальним для застосування в інших системах, платформах чи мові програмування.

При дослідженні патенту [12] було виявлено використання для оптимізації методів, які направлені на оптимізацію шляхом редагування зовнішніх характеристик сайту. Даний винахід надає системи і способи для проведення експерименту на веб-сторінці з метою оптимізації їх подання. Веб-сторінка, яка є предметом такого експерименту, за запитом браузера буде або виглядати як спочатку розроблена, або з'явиться як її варіант. Кожен варіант втілює одне або кілька змін у вихідній веб-сторінці. В ході експерименту відстежуються взаємодії з веб-сторінкою, і протягом певного періоду часу, завдяки багаторазовому перегляду вихідної веб-сторінки і її варіантів декількома особами, може бути розроблена статистика. З цієї статистики можна оцінити варіанти, які стосуються вихідної веб-сторінці і один до одного, щоб визначити, які зміни покращують уявлення і повинні бути включені в веб-сторінку.

Цей винахід також надає браузерний додаток-редактор для створення варіантів, які будуть використовуватися в експериментах. Додаток редактора може ідентифікувати, чи включає веб-сторінка короткий код, і якщо немає, запитує у сторонньої обчислювальної системи дзеркальну копію веб-сторінки, до якої доданий короткий код. Коли браузер виконує цей код після того, як веб-сторінка з кодом була повернута, код викликає подальший пошук коду, і цей код дозволяє додатку-редактору працювати в повному обсязі. При роботі додаток редактора відключає можливість взаємодії з елементами веб-сторінки звичайним способом, наприклад, клацаючи через призначений для користувача інтерфейс, і замість цього дозволяє змінювати елементи на веб-сторінці, потім записує кожну модифікацію і передає набір змін для кожного конкретного випадку назад в сторонню обчислювальну систему . Модифікації веб-сторінки, які визначають кожен варіант, додаються до коду, який повертається щоразу, коли код, доданий до веб-сторінці, виконується.

Цей винахід пропонує способи оптимізації веб-сторінок за допомогою використання експериментів, які порівнюють відповіді глядачів при поданні або з вихідної веб-сторінкою, або з її варіантом. Приблизний спосіб проведення експерименту включає отримання з першої обчислювальної системою, такий як сервер, URL-адреси веб-сторінки, яка повинна бути предметом експерименту.

Даний патент описує способи та браузерний додаток-редактор, описані принципи роботи, які відображені у вигляді схем та детального опису. Однак, невідомі результати дослідження та відсутні дані застосування даного патенту на практиці.

Досягнення підвищення продуктивності сайту залежить від методів, що надають це підвищення. Було вирішено використовувати ряд методів та технологій, що повинні

оптимізувати веб сайт, який не залежить від платформи в режимі реального часу. Серед цих застосувань було вирішено обрати наступні:

а) Content Delivery Network (CDN);

б) Кешування;

в) Мінімізація та оптимізація коду;

г) Стискання зображень.

Про кожен з обраних застосувань опишемо нижче.

1.2.1. Content Delivery Network (CDN)

Мережа доставки контенту або мережа розповсюдження контенту (CDN) - це географічно розподілена мережа проксі-серверів та їх центрів обробки даних. Мета полягає в тому, щоб забезпечити високу доступність і високу продуктивність, поширюючи службу просторово відносно кінцевих користувачів. CDN сьогодні обслуговують більшу частину Інтернет-контенту, включаючи веб-об'єкти (текст, графіку та сценарії), завантажуванні об'єкти (мультимедійні файли, програмне забезпечення, документи), програми (електронна комерція, портали), потокове мультимедіа, потокове передавання даних на вимогу ЗМІ та сайти соціальних медіа. [13]

CDN - це найпоширеніший термін, що охоплює різні типи послуг доставки контенту: потокове відео, завантаження програмного забезпечення, прискорення веб- та мобільного контенту, ліцензований / керований CDN, прозоре кешування та послуги для вимірювання продуктивності CDN, балансування навантаження, багатоканальної комутації та аналітики інтелект хмари. Постачальники CDN можуть переходити на інші галузі, такі як безпека, захист від DDoS та брандмауери веб-додатків (WAF), а також оптимізація WAN.

CDN - це мережа комп'ютерів, які доставляють контент. Більш конкретно, це безліч серверів, географічно розташованих між вихідним сервером будь-якого веб-контенту і робить запит його користувачем, їх метою є більш швидка доставка контенту за рахунок зменшення латентності.

Ці географічно ближчі сервери, також звані PoP або Points of Presence, також кешують вміст, що дозволяє видалити більшу частину завантаження контенту з вихідного сервера.

Вузли CDN зазвичай розгортаються в декількох місцях, часто за кількома магістральних лініях. Переваги включають зниження витрат на пропускну здатність, скорочення часу завантаження сторінки або збільшення доступності контенту по всьому світу. Кількість вузлів і серверів, що є CDN, варіюється в залежності від архітектури, деякі досягають тисяч вузлів з

десятками тисяч серверів у багатьох віддалених точках присутності (PoP). Інші будують глобальну мережу і мають невелику кількість географічних точок доступу.

Мережі доставки контенту доповнюють наскрізну транспортну мережу, поширюючи в ній різні інтелектуальні додатки, які залежать методи, призначені для оптимізації доставки контенту.

Запити на контент зазвичай алгоритмічно спрямовані на вузли, які в деякому роді є оптимальними. При оптимізації продуктивності можна вибрати місця розташування, які найкраще підходять для надання контенту користувачеві. Це можна виміряти, вибравши місця з найменшою кількістю переходів, найменшою кількістю мережевих секунд від запитувача клієнта або з максимальною доступністю з точки зору продуктивності сервера (як поточної, так і хронологічної), щоб оптимізувати доставку по локальних мережах. При оптимізації вартості можна вибрати місця, які є найменш дорогими. В оптимальному сценарії ці дві мети мають тенденцію до узгодження, оскільки прикордонні сервери, які знаходяться близько до кінцевого користувача на кордоні мережі, можуть мати перевагу в продуктивності або вартості.

Більшість провайдерів CDN надаватимуть свої послуги через різні, певні набори PoP, в залежності від бажаного покриття, такі як США, Міжнародний або Глобальний, Азіатсько-Тихоокеанський регіон і т. Д. Ці набори PoP можна назвати «краями», «граничні вузли" або "граничні мережі", оскільки вони будуть найближчим краєм активів CDN до кінцевого користувача.

Крайова мережа CDN зростає назовні від витоків шляхом подальших придбань (через придбання, перегляд або обмін) об'єктів спільного розміщення, пропускної здатності та серверів.

Мережі доставки контенту використовують в залежності від потреб замовника та поділяються на наступні типи застосування:

а) Контент-орієнтовані CDN. Спочатку CDN були призначені тільки для статичного контенту (JS, CSS, HTML). Ви повинні були відправляти контент, коли ви його створили / завантажували (вони не знали, що їм необхідно оновити кеш вашим контентом, навіть якщо хтось запитував його).

Потім були додані пуллінги джерела, що дозволило все автоматизувати – це означало, що, коли користувач запитував URL-адресу CDN, CDN автоматично запитував URL-адресу вихідного сайту, кешуючи контент після його повернення. Крім того, важливим фактором стала доступність. Багато CDN тепер кешують стан «останнього живого» веб-сайту, тому, якщо джерело дає збій, контент на CDN все ще доступний для користувачів, створюючи ілюзію стабільності, поки все не повернеться до нормального стану.

Крім того, сучасні CDN часто пропонують рівні автоматичної оптимізації, які автоматично змінюють розмір зображень і зберігають їх для майбутнього використання в залежності від необхідного розміру зображення. Це означає, що якщо ваш сайт містить зображення заголовка в 2 МБ, а хтось запитує його на екран шириною 300 пікселів, CDN зробить копію розміром 30 КБ і шириною 300 пікселів і буде обслуговувати її в майбутньому для всіх мобільних користувачів, автоматично роблячи сайт швидше.

б) CDN, орієнтовані на безпеку. Останнім рівнем функціоналу, доданим до CDN, став захист від DDoS і спамерських пошукових роботів. Такі CDN, як Incapsula, спеціалізуються саме на цьому. Оскільки CDN є зовнішнім шаром інфраструктури веб-сайту і першим одержувачем трафіку, він може виявляти DDoS-атаки раніше і блокувати їх за допомогою спеціальних серверів DDoS-захисту, званих скрубберами, щоб атаки вони ніколи не доходили до вихідного сервера.

Крім того, використовуючи знання, отримані від численних клієнтів, CDN може дізнатися про підозрілих IP-адреси, спамери, ботлери, навіть окремих типах сканерів і їх поведінці.

Більш того, в той час як CDN дозволяють клієнтам завантажувати свої сертифікати, вони також пропонують власні. Це можливість має дві переваги:

- коли з'являється уразливість в сертифікатах, CDN зазвичай реагують швидко, тому що вони багато втрачають (всіх своїх клієнтів). Отже, виправлення зазвичай реалізується, перш ніж більшість людей навіть дізнається про проломи в безпеці.
- більш швидке з'єднання, тому що, якщо багато веб-сайти використовують один і той же CDN, ви вже встановили коректне з'єднання і взаємна довіра з CDN через його сертифікат SSL, і цей процес не повинен повторюватися для кожного сайту, який використовує цей сертифікат CDN. Це не впливає на окремий веб-сайт так само, як на всю мережу.

До переваг CDN можна віднести:

1) Скорочення завантаження даних значно розширюється від даних користувача від сервера, тому що використовуються технології TCP / IP, які отримують маршрутизатори, які проходять через контент. Завдяки розміщенню контенту між декількома серверами CDN багаторазово знижує швидкість до кінця користувача, за рахунок чого досягається більш висока швидкість завантаження даних.

2) Застосування CDN зменшує кількість переходів, що позитивно впливає на швидкість скачування контенту, яка залишається у користувачів регулярної протягом усього часу завантаження при високій якості потоку. Це дозволяє використовувати CDN для трансляції відео-файлів в HD-форматі, здійснювати завантаження файлів великого розміру, пропонувати відео-мовлення QoS.

3) CDN мінімізує затримки даних, збої зв'язку і втрати на перевантажених каналах, на стиках між ними. CDN дозволяє управляти навантаженням магістралей і вузлів мережі, здійснюючи її розподіл між серверами.

4) Якщо розмістити сервери недалеко від кінцевих користувачів, істотно збільшиться пропускна здатність мережі: так, при наявності порту на 100 Мбіт / с її фактична швидкість може бути значно нижче, наприклад, 10 Мбіт / с. Однак при наявності 10 серверів загальна пропускна здатність може зрости до 10 × 100 Мбіт / с.

5) CDN виробляють контроль цілісності даних на всіх елементах мережі, гарантуючи доступність контенту навіть при виході з ладу віддаленого або центрального сервера, втрати зв'язку між хостами.

6) Комерційні мережі доставки і дистрибуції контенту пропонують користувачеві повну статистику, тому контент-провайдер в режимі реального часу може відстежувати всі дані про скачування, доступності та актуальності контенту в будь-якому регіоні присутності.

7) Мережі CDN містять безліч географічно розподілених платформ з широким функціоналом, спільна робота яких оптимізує обробку запитів користувачів при скачуванні даних.

8) При використанні CDN дані, розміщені на центральному сервері, копіюються на периферійні платформи, кожна з яких містить повну або часткову копію розповсюджуваного контенту. Таким чином навіть при виході з ладу центрального сервера користувач зможе отримати необхідні дані.

9) Що входить до складу платформи вузол мережі оптимізує доставку контенту, відправляючи його по найкоротшому маршруту. Довжина останнього безпосередньо залежить від топологічної або ж географічної віддаленості комп'ютера користувача від сервера - або від вартості послуги передачі трафіку інтернет-провайдера.

10) Найчастіше реалізація CDN здійснюється у вигляді кешування даних, оскільки саме воно дозволяє оптимально витрачати сполучні канали мережі і дисковий простір. Слід зазначити, що перший користувач, який звернувся на сервер провайдера, бере на себе максимум тимчасових витрат на завантаження файлів - такі ж користувачі скачують вже завантажені репліки з сервера, розташованого неподалік. Іншими словами, на серверах зберігається лише той контент, який користується максимальною популярністю і актуальністю.

11) Великі CDN нерідко складаються з великого числа розподілених вузлів, розміщених в мережах локальних інтернет-провайдерів. Більшість операторів CDN вважають найважливішою пропускну здатність сполучних каналів при мінімумі точок приєднання в регіоні присутності - але якою б не була архітектура мережі, головним призначенням CDN є прискорення завантаження даних.

Реалізація CDN зазвичай тягне за собою зміну деяких записів DNS в панелі управління реєстратора (компанія, що надає доменне ім'я). Це призводить до того, що весь трафік спочатку потрапляє на CDN. Оскільки все це відбувається за лаштунками (через IP-адреси), процес прозорий для користувача.

1.2.2. Кешування

В обчислювальній техніці, кеш є апаратний або програмний компонент, який зберігає дані, так що майбутні запити, що дані можуть бути подані швидше; дані, що зберігаються в кеші, можуть бути результатом попередніх обчислень або копією даних, що зберігаються в іншому місці. Попадання в кеш відбувається, коли запитані дані можуть бути знайдені в кеші, тоді як пропажа кешу відбувається, коли це неможливо. Хіти кеша обслуговуються читанням даних з кешу, що відбувається швидше, ніж повторний розрахунок результату або читання з більш повільного сховища даних; таким чином, чим більше запитів може бути обслужено з кешу, тим швидше працює система.[15]

Кеш - це пам'ять з більшою швидкістю доступу, призначена для прискорення доступу до даних, що містяться постійно в пам'яті з меншою швидкістю доступу (тобто, основна пам'ять). Кешування застосовується ЦПУ, жорсткими дисками, браузерами, веб-серверами, службами DNS і WINS.

Щоб бути рентабельними і забезпечувати ефективне використання даних, кеші повинні бути відносно невеликими. Проте, кеші зарекомендували себе в багатьох областях обчислювальної техніки, оскільки типові комп'ютерні програми звертаються до даних з високим ступенем локальності посилань. Такі шаблони доступу демонструють тимчасову локалізацію, де запитуються дані, які вже були запитані недавно, і просторову локалізацію, де запитуються дані, які зберігаються фізично близько до даних, які вже були запитані.

Кеш складається з набору записів. Кожен запис асоційований з елементом даних або блоком даних (невеликої частини даних), яка є копією елемента даних в основній пам'яті. Кожен запис має ідентифікатор, який часто називають тегом, що визначає відповідність між елементами даних в кеші і їх копіями в основній пам'яті.

«Веб-кеш» (або «кеш HTTP») — інформаційна технологія для тимчасового зберігання (кешування) веб-документів і зображень задля зменшення серверних затримок. Система веб-кешу зберігає копії документів, що проходять через неї; подальші запити можуть бути виконані з кешу за певних умов. Система веб-кешу може посилатися або на програмно-апаратний комплекс, або на комп'ютерну програму.[16]

Веб-браузери і веб-проксі-сервери використовують веб-кеші для зберігання попередніх відповідей від веб-серверів, таких як веб-сторінки і зображення. Веб-кеши скорочують обсяг інформації, яка повинна передаватися по мережі, оскільки інформація, раніше збережена в кеші, часто може використовуватися повторно. Це знижує пропускну здатність і вимоги до обробки веб-сервера, а також допомагає підвищити швидкість відгуку для користувачів Інтернету.

У веб-браузерах використовується вбудований веб-кеш, але деякі постачальники інтернет-послуг (ISP) або організації також використовують кешований проксі-сервер, який є веб-кешем, який використовується усіма користувачами цієї мережі.

Іншою формою кешування є P2P-кешування, коли файли, найбільш затребувані одноранговими додатками, зберігаються в кеші ISP для прискорення P2P-передачі. Аналогічно, існують децентралізовані еквіваленти, які дозволяють співтовариствам виконувати одну і ту ж задачу для трафіку P2P, наприклад, Corelli.

Кешування веб-вмісту допомагає підвищити швидкість відгуку ваших веб-сайтів за рахунок зниження навантаження на внутрішні ресурси і перевантаження мережі. Веб-кешування виконується шляхом збереження HTTP-відповідей і веб-ресурсів в кеші з метою виконання майбутніх запитів з кешу, а не з вихідних серверів.[17]

Для ефективного використання веб-кеша можуть застосовуватися різні методи веб-кешування. Самий базовий рівень - це веб-кешування на стороні клієнта, коли користувач веб-сайту використовує вбудований HTTP-кеш, вбудований в браузер. Це спрощена міра, яка може бути використана для зменшення затримки, пов'язаної із запитом веб-ресурсів з веб-сайту. Методологія кешування заснована на директивах заголовка HTTP, що надаються відповіддю HTTP від серверів походження до браузеру. Заголовки кеша HTTP надають подробиці про те, як довго браузер зможе виконувати майбутні відповіді з кеша для запитаного веб-вмісту.

На стороні сервера різні методи веб-кешування можуть бути використані для підвищення продуктивності сайту. Кеші зворотного проксі-сервера або прискорювачі веб-додатків можуть бути розміщені перед додатками і веб-серверами, щоб обслуговувати кешовану версію відповідей HTTP, збережених від них. Ці кеші реалізуються адміністраторами сайту і виступають в якості посередників між браузером і вихідними серверами. Вони також зазвичай засновані на директивах кеша HTTP.

Інша форма веб-кешування на стороні сервера включає використання сховищ ключів / значень, таких як Memcached і Redis. На відміну від зворотних проксі, які використовуються для простого кешування HTTP-відповіді на даний HTTP-запит. Сховище об'єктів ключ / значення може використовуватися для кешування будь-якого веб-контенту, необхідного розробнику додатки. Веб-контент зазвичай витягується за допомогою коду програми або використання прикладної інфраструктури, яка може використовувати сховище даних в пам'яті.

Ще однією перевагою використання сховищ ключів / значень для веб-кешування є те, що вони також зазвичай використовуються для зберігання веб-сеансів та іншого кешованого вмісту. Це забезпечує єдине рішення для різних варіантів використання.

Ефективне кешування допомагає як користувачам, так і контент-провайдерам. До основних переваг кешування можна віднести:

а) Зниження мережевих витрат. Контент можна кешувати в різних точках мережевого шляху між споживачем і джерелом контенту. Коли контент кешується ближче до споживача, запити не потребуватимуть значної мережевої активності за межами кешу.

б) Підвищення відгуку. Кешування дозволяє отримувати контент швидше, тому що немає необхідності знову пробляти шлях по всій мережі. Кеші, підтримувані поруч з користувачем, наприклад кеш браузера, можуть зробити обслуговування запиту майже миттєвим.

в) Підвищена продуктивність. Контент-провайдер може використовувати потужні сервери в шляху доставки, щоб взяти на себе основне навантаження на обслуговування контенту.

г) Доступність контенту під час збоїв мережі. При використанні певних політик кешування може обслуговувати контент користувачам навіть протягом коротких збоїв.

Деякий контент легше кешувати. Для більшості сайтів кешувати краще:

- Логотипи та зображення бренду.
- Не змінювані зображення в цілому (наприклад, значки навігації).
- Стилі.
- Загальні файли Javascript.
- Завантаження контент.
- Файли мультимедіа.

Ці елементи змінюються нечасто, тому їх можна кешувати протягом більш тривалих періодів часу.

Також існують елементи, що потрібно кешувати обережно, серед яких:

- HTML-сторінки.
- Зображення, що часто змінюються.
- Часто змінювані JavaScript і CSS.
- Контент, запитуваний за допомогою файлів cookie.

Окрім цього є ряд елементів, що кешувати не рекомендується:

- Активи, пов'язані з конфіденційними даними. (банківська інформація тощо)
- Контент, який залежить від користувача і часто змінюється.

На додаток до вищевказаних загальними правилами можна налаштувати політики, які дозволять вам відповідним чином кешувати різні типи контенту. Наприклад, якщо всі авторизовані користувачі бачать один і той же вид сайту, може бути корисно кешувати його. Якщо авторизовані користувачі бачать призначений для користувача вигляд сайту, який буде дійсний протягом деякого часу, ви можете зберегти кеш в браузері користувача, але виключити проміжне кешування.

Є кілька кроків, які можна зробити, щоб збільшити коефіцієнт попадання в кеш, перш ніж звернутися до заголовків:

- встановлення певних каталогів для зображень, css і загального контенту. Розміщення контенту в виділених каталогах дозволить вам легко посилатися на них з будь-якої сторінки вашого сайту.
- використання одного і того ж URL-адресу для посилання на одні й ті ж елементи. Оскільки кеш відштовхується як від хоста, так і від шляху до запрошенням контенту, переконайтеся, що ви посилаєтеся на свій контент на всіх ваших сторінках однаково.
- використання CSS-спрайт, де це можливо. CSS-спрайт для елементів, таких як значки і навігація, зменшують кількість раундів, необхідних для рендеринга вашого сайту, і дозволяють сайту кешувати цей єдиний спрайт протягом тривалого часу.
- по можливості зберігати сценарії і зовнішні ресурси локально. Якщо ви використовуєте скрипти JavaScript і інші зовнішні ресурси, подумайте про те, щоб розмістити ці ресурси на своїх власних серверах. Зверніть увагу, що вам потрібно знати про будь-яких оновленнях, щоб вчасно оновити локальну копію.

Головною метою політики кешування є досягнення балансу, який дозволяє застосовувати агресивну кешування, коли це можливо, і залишає можливість анулювати записи при внесенні змін. Швидше за все, на кожному сайті будуть:

- Агресивно кешувальні елементи.
- Елементи з коротким терміном свіжості і необхідністю валідації.
- Елементи, що не кешуються взагалі.

Мета полягає в тому, щоб по можливості переміщати контент в перші категорії при збереженні прийнятного рівня точності.

1.2.3 Мінімізація та оптимізація коду

Оптимізація коду - різні методи перетворення коду заради поліпшення його характеристик і підвищення ефективності. Серед цілей оптимізації можна вказати зменшення обсягу коду, обсягу використовуваної програмою оперативної пам'яті, прискорення роботи програми, зменшення кількості операцій введення виведення.[18]

Головне з вимог, які звичайно пред'являються до методу оптимізації - оптимізована програма повинна мати той же результат і побічні ефекти на тому ж наборі вхідних даних, що і неоптимізованими програма. Втім, ця вимога може і не грати особливої ролі, якщо виграш за рахунок використання оптимізації може бути визнаний більш важливим, ніж наслідки від зміни поведінки програми.

Оптимізація коду може проводитися як і вручну, програмістом, так і автоматизовано. В останньому випадку оптимізатор може бути як окремим програмним засобом, так і бути вбудованим в компілятор (так званий оптимізуючий компілятор). Крім того, слід зазначити, що сучасні процесори можуть оптимізувати порядок виконання інструкцій коду.

Існують такі поняття як високорівнева і низькорівнева оптимізація. Високорівневі оптимізації в більшості проводяться програмістом, який, оперуючи абстрактними сутностями (функціями, процедурами, класами і т.д.) і уявляючи собі загальну модель вирішення задачі, може оптимізувати дизайн системи. Оптимізації на рівні елементарних структурних блоків вихідного коду (циклів, розгалужень і т.д.) теж зазвичай відносять до високого рівня; деякі виділяють їх в окремий ("середній") рівень. Низькорівнева оптимізація проводиться на етапі перетворення вихідного коду в набір машинних команд, і найчастіше саме цей етап піддається автоматизації. Втім, програмісти на асемблері вважають, що ніяка машина не перевершить в цьому хорошого програміста (при цьому всі згодні, що поганий програміст зробить ще гірше і машини).

При оптимізації коду вручну існує ще одна проблема: потрібно знати не тільки, яким чином проводити оптимізацію, але і в якому місці її застосувати. Зазвичай через різні факторів (повільні операції введення, різниця в швидкості роботи людини-оператора і машини і т.д.) лише 10% коду займають цілих 90% часу виконання (звичайно, твердження досить умоглядно, і має сумнівне підставу у вигляді закону Парето, проте виглядає досить переконливо у Е. Таненбаума). Так як на оптимізацію доведеться витрачати додатковий час, тому замість спроб оптимізації всієї програми краще буде оптимізувати ці "критичні" до часу виконання 10%. Такий фрагмент коду називають вузьким місцем або пляшковим горлечком (bottleneck), і для його визначення використовують спеціальні програми - профайлери, які дозволяють заміряти час роботи різних частин програми.

1.2.4. Методи стиснення зображень

Стиснення зображень - застосування алгоритмів стиснення даних до зображень, що зберігаються в цифровому вигляді. В результаті стиснення зменшується розмір зображення, через що зменшується час передачі зображення по мережі і економиться простір для зберігання.

Стиснення зображень поділяють на стиск з втратами якості і стиснення без втрат. Стиснення без втрат часто краще для штучно побудованих зображень, таких як графіки, іконки програм, або для спеціальних випадків, наприклад, якщо зображення призначені для подальшої обробки алгоритмами розпізнавання зображень. Алгоритми стиснення з втратами при збільшенні ступеня стиснення як правило породжують добре помітні для людського ока артефакти [20].

При стисненні зображень необхідно дотримуватися наступних умовних правил:

а) Стискати зображення в потрібному форматі в найменшому прийнятній якості:

1) Налаштування рівня стиснення всіх зображень вручну, де це можливо.

2) Автоматизувати стиснення інших, щоб досягти найвищої продуктивності.

б) Перевіряти можливість використання формату WebP для зображень;

в) Необхідно зберігати зображення з прогресивним налаштуванням. Це сприятливо позначиться на сприйнятті користувачами сайту;

г) Дослідження інших способів досягнення кращого стиснення або прозорості.

Якщо говорити дуже спрощено, чим більше сторінка, тим довше вона завантажується. Проведено безліч досліджень, які показують, що користувачі повільних сайтів проводять менше часу на сайті, менше натискають на посилання і рекламні блоки, і витрачають менше грошей. Маленькі сайти, такі як AutoAnything, які вдвічі скоротили час завантаження сайту, відзначили збільшення виручки на 13 відсотків. А дослідження на великих сайтах, таких як Amazon, показали, що їх виручка знижується на 1 відсоток на кожні 100 мілісекунд затримки завантаження.

До того ж до того, що великі сторінки негативно впливають на продуктивність стаціонарних комп'ютерів, найбільше роздуті сторінки змушують страждати мобільних користувачів. Сторінка розміром в 1 Мб не тільки нескінченно довго завантажується, вона також може неприємно позначитися на рахунку за послуги зв'язку.

Навіть так звані «безлімітні» тарифні плани для мобільних телефонів, насправді такими не є. Більшість з них надають близько 2 Гб за фіксовану плату, а перевищення цього ліміту може тарифікуватися додатково. Не кажучи вже про те, що є безліч місць в світі, де відсутні безлімітні тарифні плани і де вартість завантаження інформації є серйозною проблемою для користувачів.

Ми можемо бачити, що найбільшу частку в обсязі сторінки займають зображення. Це погані новини для мобільних пристроїв, де картинка буквально коштує тисячі слів.

Неправильна обробка зображень часто стає причиною поганої продуктивності. Занадто великі, недостатньо стислі зображення або зображення дуже високої якості можуть виявитися занадто важкими, що має прямий вплив на швидкість завантаження вашого сайту. Вибрати правильний метод стиснення і досягти при цьому найкращих результатів нескладно, знаючи про особливості цього процесу.

Формат зображень для забезпечення економії розміру при стисненні, як правило, з'єднує разом різні алгоритми стиснення з втратами і без. Існує декілька форматів, підтримуваних веб браузерами, з різними характеристиками. Точніше, не існує (поки) одного формату «на всі випадки життя». Різні види зображень повинні бути закодовані в різні формати, в залежності від того, що це за зображення, від того, які формати підтримує браузер і від потреб конкретної веб-сторінки.

Налаштування параметрів кожного зображення на вашій сторінці, що проводиться з метою збалансувати якість і розмір, дозволить вам домогтися найкращої економії при найкращих рівнях якості. Однак на великих сайтах розміщено безліч зображень, і вони зазвичай не можуть вручну оптимізувати кожне, тому індивідуальна для кожного зображення настройка рівня якості практично неможлива.

Деякі розробники вибрали більш автоматизований підхід до цього типу кодування, часто застосовуючи до зображень власні евристичні процеси і процеси кодування під час розробки. Цей спосіб обробки підходить в якості золотої середини між налаштуванням і автоматизацією, яка допоможе більшості веб-розробників. Можна також використовувати різні додатки, наприклад JPEGmini, яке також автоматично налаштує рівень стиснення JPG, щоб отримати найкращу якість.

Трохи радикальним підходом, який використовується розробниками, щоб зменшити розмір зображення, є перетворення всіх простих значків і зображень у файли SVG, і їх растеризація на стороні клієнта перед відображенням.

Цей процес - компроміс між розміром файлу та швидкодією клієнта: він економить трафік, але провокує велике навантаження на стороні клієнта, щоб реконструювати зображення при рендеринга.

Як такий, формат зображень SVG сильно відрізняється від інших типів файлів тим, що це векторний формат, тобто остаточне зображення генерується процедурно, використовуючи інформацію про описані в файлі формах, до отримання певного остаточного зображення.

Також, так як і для ресурсів, є можливість застосування GZip, але тільки для SVG та JPEG форматів. На жаль, для стиснення форматів PNG та GIF необхідно використовувати

сторонні засоби та розміщувати уже стиснені файли. Серед них можна виділити наступні популярні застосування:

а) Iloveimg.com [21] – російськомовний онлайн сервіс для стиснення та іншої обробки фото з забором фото з диска комп'ютера і двох хмарних серверів.

1) Відмінний компресор допомагає швидко стиснути JPG, PNG або GIF фото зі збереженням якості.
2) Можна забрати фото з Google Drive і Dropbox.
3) Крім цього: конвертує фото в / з JPG, повертає, обрізає, змінює розмір.

б) IMGonline [22] – може стиснути фото в форматах BMP, GIF, JPEG, PNG, TIFF в формат JEG по заданих налаштувань. На сервері є стандартний набір інструментів для роботи з фото:

1) Змінити розмір картинки;
2) Конвертувати формат;
3) Редактор мета даних EXIF;
4) Ефекти для фото (20 штук);
5) Поліпшити якість фото;
6) І ще близько 30 фото інструментів.

г) JPG Zipper [23] – відмінний інструмент онлайн стиснення фотографій формату JPG. Працює моментально, без завантажувача, тільки перетягування фото.

В результаті проведеного аналізу зроблені наступні висновки про напрям дослідження і перспективи, подальший розвиток у даному напрямку:

а) на даний момент існує багато різних методів та технологій оптимізації сайтів, проведено ряд розробок науковими діячами, що забезпечують швидку роботу, продуктивність та підвищує рейтинги сайту в пошукових системах, однак майже всі описані роботи лише інформують та рекомендують ці методи, не надаючи результатів дослідження;

Б) застосування методів оптимізації має бути усвідомленим, оскільки застосування того чи іншого методів може погіршити SEO-оптимізацію або навіть зменшити швидкість роботи сайту.

1.3 Постановка наукової задачі та обґрунтування методики досліджень

Результати проведеного аналізу методів та технологій крос-платформної оптимізації веб-сайтів показали, що на сьогодні аналогічних розробок не багато, оскільки вони не мають практичного застосування. Аналоги мають кожна з поставлених задач як окреме рішення, що не

є оптимальним, оскільки вирішення однієї задачі може викликати проблему при виконанні інших.

В оброблених публікаціях надаються рекомендації щодо використання тих чи інших методах, в залежності від направлення оптимізації. Кожна стаття добре описує існуючі методи та в яких випадках їх використовувати. Однак, кожна з представлених публікацій не надає даних про практичне застосування методів і технологій та має лише інформативну цінність.

На основі проаналізованої інформації було поставлені наступні завдання:

1) підвищення продуктивності веб-сайту за допомогою методу удосконалення крос-платформної оптимізацію сайту;

2) відобразити результати дослідження у вигляді дисертаційної роботи;

3) практично реалізувати запропонований метод та провести порівняльний аналіз з існуючими методами на основі вихідного та оптимізованого веб-сайтів.

Дослідження доцільно проводити з застосуванням наступних технологій: CDN, кешування, стиснення зображень, мінімізації та оптимізації коду, що відноситься саме до серверної та клієнтської оптимізації.

1.4 Висновки до першого розділу

В процесі дослідження виявлено закономірність наявності схожих робіт з даною темою, однак кожна з представлених публікацій не мала на меті розробити оптимізований сайт, а лише надавала рекомендації по використанню цих методів.

Результатом аналізу було виявлено наявність прямих та непрямих методів та технологій, які мають спосіб оптимізації в залежності від направлення оптимізації.

Запропоноване рішення було отримано, на основі аналізу непрямих методів та технологій, та полягає в удосконаленні методів оптимізації шляхом об’єднання серверної та клієнтської технологій.

Область застосування даних технологій в середовищі веб-розробок, оскільки розроблені методи можна використовувати при створенні веб-сайтів будь-якої складності, платформи чи способу його створення.

Наведено детальний опис засобів крос-платформної оптимізації сайтів.

Мережа доставки контенту, або CDN, показала себе однією з найважливіших технологій для досягнення мети даної дисертаційної роботи, оскільки її використання має великий вплив на підвищення продуктивності. Окрім цього, CDN включає метод кешування, що також може замінити використання звичайного кешування або взагалі відмовитись від нього. Технологія

має великий ряд переваг, які пов’язані з підвищенням продуктивності, що підтверджує рішення її використання.

Використання кешування також виправдане, оскільки його застосування хоча і не підвищує завантаження сайту при першому переході на нього, однак швидкість завантаження в наступні рази декілька підвищується.

Мінімізація та оптимізація коду – один з найголовніших атрибутів сучасного сайту. Все це пояснюється тим, що використання цих методів необхідне не тільки для підвищення швидкості роботи, але й підвищення показників для пошукової оптимізації.

Визначено, що найефективнішою технологією, що використовується для крос-платформної оптимізації сайту є стискання зображень, оскільки при стисканні підвищується швидкість завантаження сайту навіть при першому запуску.

РОЗДІЛ 2
АНАЛІТИЧНИЙ ОГЛЯД

Концепція запропонованого методу удосконалення крос-платформної оптимізації має на увазі застосування наступних технологій у визначеній послідовності, як це зазначено на рис. 2.1.

Рисунок 2.1 – Метод крос-платформної оптимізації

На першому етапі застосовується технологія Content Delivery Network (CDN), використання якої передбачає застосування ресурсу Clodflare. Використання цієї технології дозволяє скоротити витрати і час транзиту при високій швидкості доставки даних. Далі використовується кешування, що визначає три основні механізми керування кешами: свіжість, валідація та анулювання. На наступному етапі застосовується технологія мінімізації та оптимізації коду з використанням мініфікації та методів, що лежать в основі ресурсу Gzip. Далі застосовується технологія стиснення зображень з використанням алгоритму JPEG. Детально технології розглянуто в наступних розділах.

2.1 Технологія Content Delivery Network (CDN)

Розглянемо використання технології CDN на першому етапі удосконалення крос-платформної оптимізації веб-сайту.

Існують різні типи CDN, що пропонують різні види послуг, вони можуть мати різну топологію мережі: розподілені CDN прагнуть мати якомога більше серверів по всьому світу. Консолідовані CDN мають менше точок, але більше налаштовані на збільшення продуктивності мережі, пропускної спроможності і протидія DDoS.

При використанні засобів CDN поширення контенту відрізняється від звичайного одно-точкового (див. рис. 2.2, рис. 2.3).

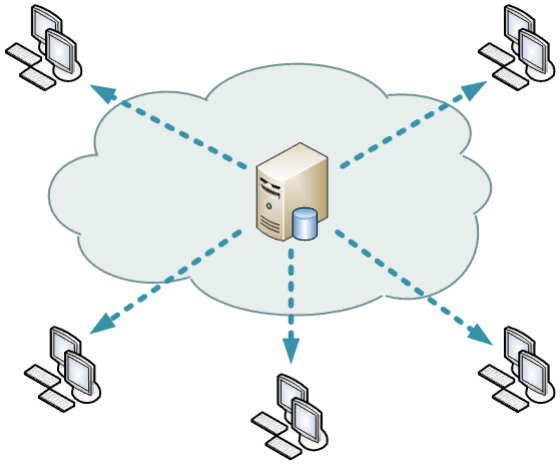


Рисунок 2.2 – Одноточкове поширення контенту

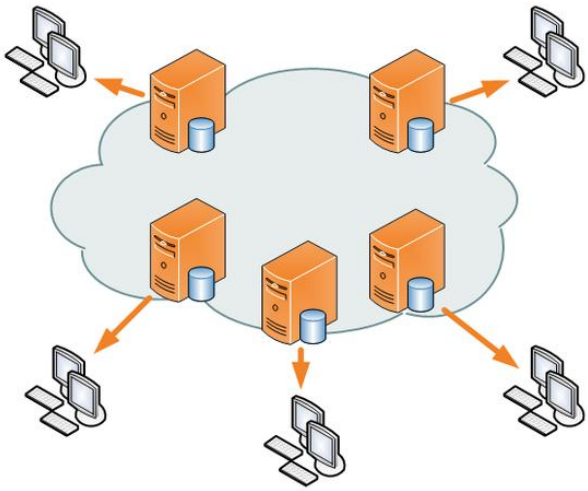


Рисунок 2.3 – Поширення контенту засобами CDN

При розробці методу використано веб-сервіс Cloudflare. Cloudflare – одна з провідних сучасних мереж доставки контенту. Сервери цієї фірми розташовані в стратегічно обраних дата центрах по всьому світу, щоб прискорити доставку статичних файлів відвідувачам вашого сайту.

Cloudflare створили глобальну мережу, призначену для оптимізації безпеки, продуктивності та надійності, без наважої технології. Cloudflare CDN легко налаштовується, є більш доступним і працює краще, ніж будь-який інший CDN.

Cloudflare розгортає менші сліди на більшій кількості місць. Це дозволяє нам скористатися перевагами інноваційних стратегій розгортання, живлення та охолодження[24]. Як результат, сервіс зміг оптимізувати обробку великих обсягів трафіку через свою мережу.

За замовчуванням Cloudflare кешує лише статичні файли. Тим не менш, правила сторінок можуть бути використані для встановлення більшої кількості файлів, які можна кешувати. За допомогою функцій підприємства, таких як сортування рядків запитів, Cloudflare спочатку сортуватиме рядки запитів у URL-адресі в детермінованому порядку, перш ніж перевіряти кеш для ресурсу або запитувати його з початку. Ця функція особливо корисна для серверів API, які часто покладаються на кілька аргументів рядка запитів.

Завдяки вбудованій системі балансування навантаження та автоматичному відновленню, мережа Cloudflare була розроблена, щоб витримати 50% мережі, не впливаючи на доступність послуг.

За своєю суттю, CDN - це мережа серверів, з'єднаних разом з метою швидкого, дешевого та надійного надання контенту. Щоб покращити швидкість і підключення, CDN розмістить сервери в точках обміну між різними мережами. Ці пункти обміну в Інтернеті (IXP) є основними місцями, де різні провайдери Інтернету підключаються, щоб надати один одному доступ до трафіку, що походить з їхніх різних мереж. Завдяки підключенню до цих високошвидкісних і взаємопов'язаних місць постачальник CDN здатний скоротити витрати і час транзиту при високій швидкості доставки даних (див. рис.2.4).

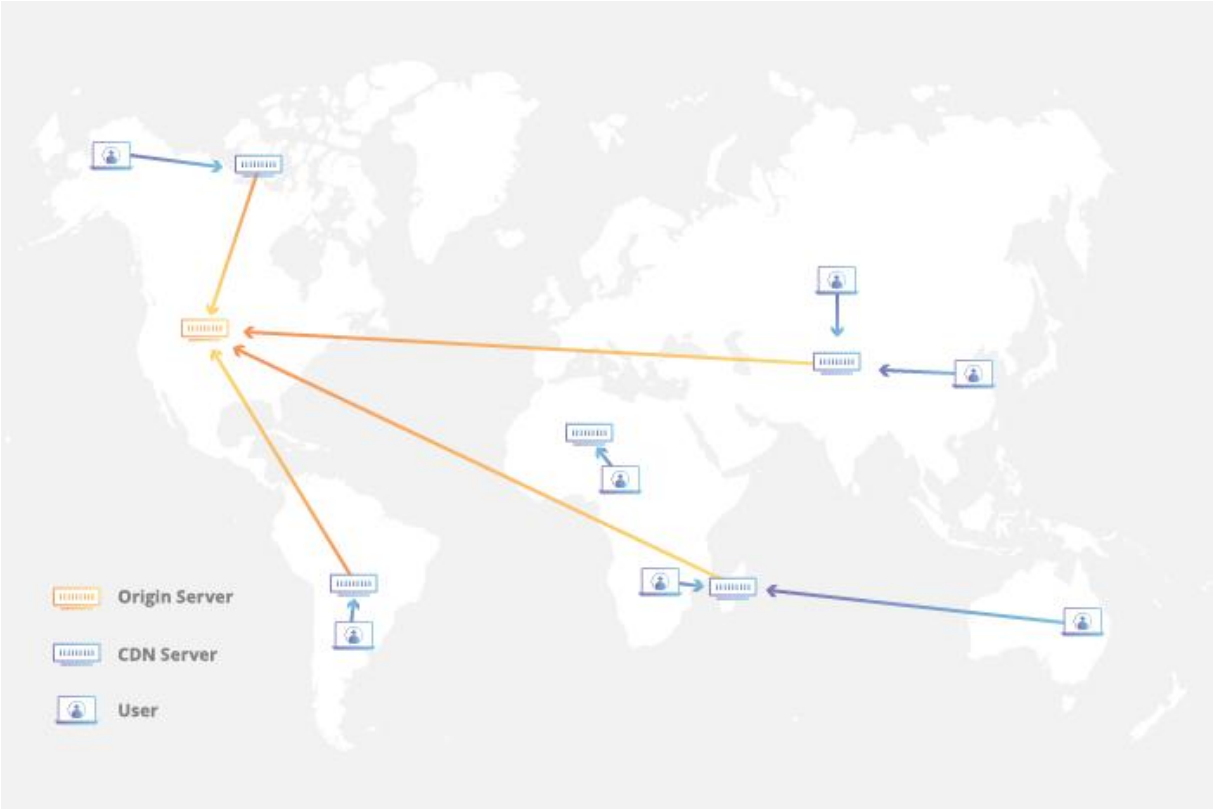


Рисунок 2.4 – Алгоритм роботи CDN

Cloudflare CDN за допомогою механізму зворотного проксінг nginx роздає сайти. Принцип роботи простий: у них розміщений дуже швидкий веб-сервер nginx який першим приймає на себе користувачів, і далі вже самостійно звертається до ваших ресурсів. При цьому по шляху відсікає вірусний трафік, типу ДДОС-атак і перевіряє на наявність запитуваної сторінки свій кеш. Якщо сторінка там є, і вона не змінилася, то він просто віддає її, навіть не звертаючись до основного сервера.

2.2 Технологія кешування

Наступний етап удосконалення крос-платформної оптимізації веб-сайту полягає у використанні кешування.

Веб-кеш - це виділена комп'ютерна система, яка буде відслідковувати запити об'єктів і зберігати об'єкти при їх отриманні з сервера. При повторних запитів кеш буде доставляти об'єкти зі свого сховища, а не передавати запит на вихідний сервер. Кожен веб-об'єкт змінюється з часом і тому має термін життя або «свіжість». Якщо свіжість об'єкта закінчується, відповідальність за отримання нової версії об'єкта лежить на веб-кеші. Чим більше запитів до одного і того ж об'єкту, тим ефективніше буде веб-кеш при скороченні висхідного трафіку, що

також допоможе знизити навантаження на сервер, що приведе до меншої затримки.(див. рис. 2.5)

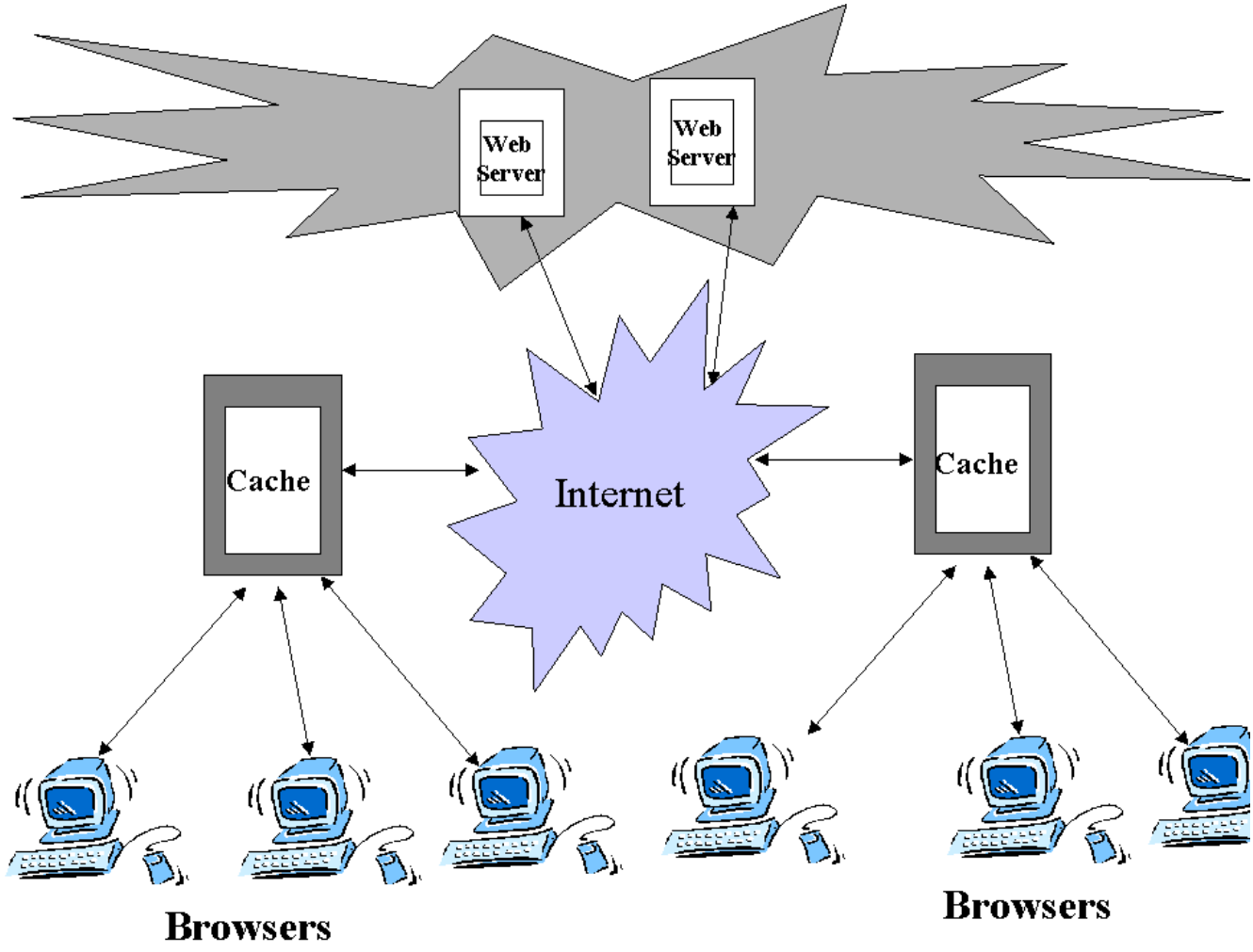


Рисунок 2.5 – Схема роботи веб-кешування

HTTP визначає три основні механізми керування кешами: свіжість, валідація та анулювання [16].

Свіжість дозволяє використовувати відповідь без повторної перевірки на вихідному сервері і може контролюватися як сервером, так і клієнтом. Наприклад, в заголовку відповіді `Expires` вказується дата, коли документ стає застарілим, а директива `Cache-Control: max-age` повідомляє кешу, наскільки секунд потрібна нова відповідь.

Одного разу потрапивши в кеш, ресурс, теоретично, може зберігатися там вічно. Однак, оскільки обсяг сховища кінцевий, записи періодично доводиться звідти видаляти. Цей процес називають витісненням даних з кеша (`cache eviction`). Крім того, на сервері ресурси можуть змінюватися, тому кеш потрібно оновлювати. Оскільки HTTP є клієнт-серверним протоколом,

сервера не можуть самі звертатися до кешам і клієнтам при зміні ресурсу; їм необхідно домовитися про термін дії збереженої копії. До його закінчення ресурс вважається свіжим (fresh), після - застарілим (stale). Алгоритми витіснення віддають перевагу "свіжим" ресурсам. Проте, копія ресурсу не видаляється з кеша відразу ж по закінченню її терміну дії. При отриманні запиту на відповідний ресурс кеш випереджає його з заголовком If-None-Match на випадок, якщо його копія все ще актуальна. Якщо це так, сервер повертає заголовок 304 (Not Modified - Чи не змінений), а тіло ресурсу не посилає, економлячи тим самим трафік.

Час старіння (freshnessLifetime) обчислюється на підставі декількох заголовків. Якщо заданий заголовок "Cache-control: max-age = N", то час старіння дорівнює N. Якщо його немає, а це буває дуже часто, перевіряється заголовок Expires, і, якщо він є, то час старіння береться рівним значенню заголовка Expires мінус значення заголовка Date. Нарешті, якщо немає ні того ні іншого, дивляться заголовок Last-Modified. Якщо він є, то час старіння дорівнює значенню заголовка Date мінус значення заголовка Last-modified розділити на 10.

Термін дії обчислюється наступним чином:

expirationTime (термін дії) = responseTime + freshnessLifetime - currentAge (поточний вік),

де responseTime - це час отримання відповіді по годиннику браузера.

Валідація може бути використана для перевірки того, чи кешована відповідь все ще гарна після того, як вона стане старою. Наприклад, якщо відповідь має заголовок Last-Modified, кеш може зробити умовний запит, використовуючи заголовок If-Modified-Since, щоб побачити, чи змінився. Механізм ETag (сутності тегів) також дозволяє як сильну, так і слабку перевірку.

Валідація кеша запускається при натисканні користувачем кнопки перезавантаження. Крім того, вона може виконуватися в ході звичайного режиму перегляду, якщо кеш відповідь включає заголовок "Cache-control: must-revalidate". Іншим фактором є налаштування кешування (вікно Advanced-> Cache preferences) - можна вимагати примусової валідації при кожному завантаженні документа.

При закінченні терміну придатності документа він або проходить валідацію, або повторно доставляється з сервера. Валідація може виконуватися тільки якщо у сервера є сильний або слабкий валідатор.

Заголовок відповіді ETag є непрозорим для агента користувача значенням, яке можна використовувати в якості сильного валідатора. Агент користувача, наприклад, браузер, не знає, що являє цей рядок і не може передбачити, яким буде її значення. Якщо заголовок ETag був

частиною відповіді, клієнт може вивести `If-None-Match` в заголовок майбутніх запитів для валідації кеш ресурсів.

Заголовок відповіді `Last-Modified` можна використовувати в якості слабкого валідатора. Слабким він вважається через те, що має 1-секундний дозвіл. Якщо у відповіді присутній заголовок `Last-Modified`, то для валідації кешованого документа клієнт може виводити в запитах заголовок `If-Modified-Since`.

При запиті на валідацію сервер може або проігнорувати його і послати стандартну відповідь 200 OK, або повернути відповідь 304 `Not Modified` (з порожнім тілом), вказуючи, таким чином, щоб браузер взяв копію з кеша. В останньому випадку у відповідь можуть входити також заголовки для поновлення строку дії кеш ресурсу.

Інвалідація, як правило, є побічним ефектом іншого запиту, який проходить через кеш. Наприклад, якщо URL-адреса, пов'язана з кешованою відповіддю, згодом отримає запит POST, PUT або DELETE, кешована відповідь буде визнана недійсною.

Найбільш гнучкий з варіантів інвалідації елементів кешу, що мають складні залежності - це теги. Тег є залежність ключа кеша від чого-небудь, виражається зазвичай довільній рядком (як і сам ключ), і дозволяє разом видаляти з кеша всі елементи, на які він встановлений.

При цьому, в принципі, можна зробити, щоб і сам тег міг залежати від інших тегів - це і буде відображати весь граф залежностей між об'єктами.

Теги є більш просунутим функціоналом, ніж TTL, і доступні далеко не скрізь. Однак їх теж можна реалізувати на існуючому кешу, з одним застереженням - метадані дуже бажано зберігати не в самому кеша.

Щоб отримати переваги інвалідації, заснованої на тегах, вам потрібно приєднати до кожного кеш об'єкту правильні теги. Кожен тег - це простий ідентифікатор рядка, який ви можете використовувати в будь-який час, щоб запустити видалення всіх об'єктів, пов'язаних з цим тегом.

Ефективне кешування допомагає як користувачам, так і контент-провайдерам. До основних переваг кешування можна віднести:

а) Зниження мережевих витрат. Контент можна кешувати в різних точках мережевого шляху між споживачем і джерелом контенту. Коли контент кешується ближче до споживача, запити не потребуватимуть значної мережевої активності за межами кешу.

б) Підвищення відгуку. Кешування дозволяє отримувати контент швидше, тому що немає необхідності знову пробляти шлях по всій мережі. Кеші, підтримувані поруч з користувачем, наприклад кеш браузера, можуть зробити обслуговування запиту майже миттєвим.

в) Підвищена продуктивність. Контент-провайдер може використовувати потужні сервери в шляху доставки, щоб взяти на себе основне навантаження на обслуговування контенту.

г) Доступність контенту під час збоїв мережі. При використанні певних політик кешування може обслуговувати контент користувачам навіть протягом коротких збоїв.

Деякий контент легше кешувати. Для більшості сайтів, як і для запропонованого, кешувати краще:

- Логотипи та зображення бренду.
- Не змінювані зображення в цілому (наприклад, значки навігації).
- Стилі.
- Загальні файли Javascript.
- Завантаження контент.
- Файли мультимедіа.

Ці елементи змінюються нечасто, тому їх можна кешувати протягом більш тривалих періодів часу.

Також існують елементи, що кешувались обережно, серед яких:

- HTML-сторінки.
- Зображення, що часто змінюються.
- Часто змінювані JavaScript і CSS.
- Контент, запитуваний за допомогою файлів cookie.

Окрім цього є ряд елементів, що не кешуються:

- Активи, пов'язані з конфіденційними даними. (банківська інформація тощо)
- Контент, який залежить від користувача і часто змінюється.

Застосовуються декілька кроків, які можна зробити, щоб збільшити коефіцієнт попадання в кеш, перш ніж звернутися до заголовків:

- встановлення певних каталогів для зображень, css і загального контенту. Розміщення контенту в виділених каталогах дозволить вам легко посилатися на них з будь-якої сторінки вашого сайту.
- використання одного і того ж URL-адресу для посилання на одні й ті ж елементи. зберігання сценаріїв і зовнішні ресурси локально.

Головною метою політики кешування є досягнення балансу, який дозволяє застосовувати агресивну кешування, коли це можливо, і залишає можливість анулювати записи при внесенні змін.

На запропонованому сайту будуть використано наступні елементи:

- Агресивно кешувальні елементи.

- Елементи з коротким терміном свіжості і необхідністю валідації.
- Елементи, що не кешуються взагалі.

За мету поставлено по можливості переміщати контент в перші категорії при збереженні прийнятного рівня точності.

2.3 Технологія мінімізації та оптимізації коду

Подальшим кроком запропонованого методу удосконалення крос-платформної оптимізації веб-сайтів є мінімізація та оптимізація коду.

Оптимізація сайтів підрозділяється на зовнішню і внутрішню. Тоді як зовнішня оптимізація спрямована на роботу із зовнішніми факторами пошукового просування сайтів, внутрішня оптимізація сайтів включає в себе оптимізацію посилальної структури сайтів, оптимізацію HTML коду і оптимізацію тексту сторінок сайтів так, щоб позиції сайтів в рейтингах пошукових систем стали якомога вище.

Оптимізацію коду можна розділити на 2 складові.

а) Комутація - зменшення навантаження на сервер і збільшення швидкості сайту. Системи управління динамічними сайтами пишуться на php. Це означає, що при формуванні сторінки і кожного її елемента відправляється запит на сервер, відповідь з якого і виводиться на екран.

Завдання полягає зменшити, по можливості, кількість комутацій між користувачем (браузер) і сервером. Зробити це можна перетворивши деякі елементи в статичні, тобто розмістити їх відразу на сторінці, щоб завантажувалися разом з нею.

б) Стиснення - прискорення обробки коду та збільшення швидкості сайту. Style.css часом роздувається до величезних розмірів, все залежить від фантазії веб майстра, і його бажання зробити свій сайт неповторним. Кожен рядок цього файлу вимагає деякого часу, який необхідний для прочитання її браузером, якщо включений його кеш, або якщо кеш не включений - час на виконання запиту.

Мініфікація CSS, JS, HTML файлів включає в себе видалення будь-яких непотрібних символів з файлу, щоб зменшити його розмір і тим самим прискорити завантаження.

Під час мініфікації файлу видаляється:

- Символи пробілів;
- Коментарі;
- Розриви рядків;
- Роздільники блоків.

У більшості випадків процес мініфікації не впливає на файл, а оптимізує його для завантаження. Особливо корисна мініфікація CSS, JS і HTML-файлів. Крім цього Google при ранжируванні враховує швидкодію ресурсу, а мініфікація допомагає прискорити роботу сайту.

Щоб розрізняти мініфікаційні файли, в їх імена додається розширення .min (наприклад: foobar.min.css).

В процесі дослідження було обрано веб сервіс minifier.org, що видаляє пробіли, видаляє коментарі, поєднує файли (у тому числі @import оператори і невеликі активи в CSS-файли) і оптимізує / скорочує кілька загальних шаблонів програмування.

Всі сучасні мініфікатори JavaScript-коду працюють таким чином:

1) Розбирають JavaScript-код у синтаксичне дерево.

2) Також надходить будь інтерпретатор JavaScript перед тим, як його виконувати. Але потім, замість виконання коду бігають по цьому дереву, аналізують і оптимізують його.

3) Записують з синтаксичного дерева вийшов код.

Мініфікація CSS файлів полягає у видаленні зайвих пробілів та коментарів цих файлів.

Мініфікація коду призводить до зменшення розміру файлів (див. рис. 2.6), використовуючи описані алгоритми, чим саме зменшує обсяг даних, які повинні бути оброблені.

Рисунок 2.6 – Мініфікація CSS файлу

Мініфікація і стиснення CSS файлів - це не одне і те ж. Хоча обидва цих методу призначені для зменшення часу завантаження. Різниця полягає в тому, як вони працюють.

Стиснення використовується для зменшення розміру файлу за допомогою алгоритмів стиснення, таких як Gzip або brotli. Файли стискаються перед відправкою клієнту.

Підтримувані алгоритми стиснення можуть варіюватися в залежності від сервера, а також браузерів. Коли браузер відправляє запит серверу, він повідомляє йому, який метод стиснення підтримує, так що сервер може оптимізувати відгук для цього браузера.

Сучасні браузери підтримують стислий контент, тому що він є частиною специфікації протоколу HTTP 1.1. Стиснення текстових форматів (CSS, Javascipt і HTML) може зменшити їх обсяг на 70% (див. рис. 2.7). Працює все дуже просто. Перед відправкою відповіді сервер стискає дані. Браузер при отриманні стисненого відповіді розтискає його і показує результат.

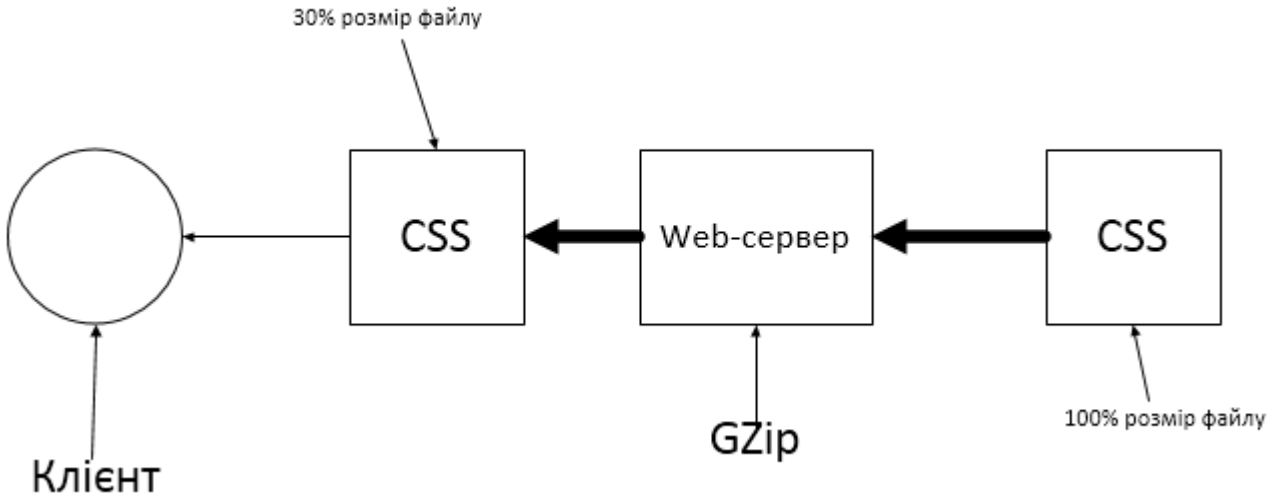


Рисунок 2.7 – Алгоритм стиснення текстових форматів утилітою GZip

Здійснення GZip стиснення виявилося можливим завдяки активації модуля `mod deflate`, який можна включити через директиви в конфігураційному файлі .htaccess, що регулює всі процеси щодо сайтів на серверах Apache.

Отже, процес стиснення здійснюється наступним чином:

- Файл стискається за допомогою алгоритму стиснення;
- Виконується запит для стислій версії файлу;
- Стиснутий файл відправляється від сервера до клієнта;
- Клієнт розпаковує файл і зчитує інформацію.

2.4 Технологія стиснення зображень

Заключним етапом запропонованого методу удосконалення крос-платформної оптимізації веб-сайтів є стиснення зображень.

Для реалізації стиснення зображень використаний ресурс Imgonline.org, який використовує алгоритм стиснення JPEG. Він має власний алгоритм роботи. При стисненні зображення перетворюється з RGB колірного простору на YCbCr. Після перетворення RGB-> YCbCr для каналів зображення Cb і Cr, які відповідають за колір, можна виконувати "витончення", яке полягає в тому, що кожному блоку 4 пікселя (2x2) каналу яскравості Y призначається середні значення Cb і Cr (схема витончення «4: 2: 0»). Крім того, для кожного блоку 2x2 замість 12 значень (4 Y, 4 Cb і 4 Cr) використовуються тільки 6 (4 Y і один усереднений Cb і Cr кожен). Якщо на якість зображення, відновленого після стиснення, пред'являються більш високі вимоги до якості, проріджування можна виконувати тільки в одному напрямку - вертикально (4: 4: 0) або горизонтально (4: 2: 2) або зовсім не (4 : 4: 4).

Стандарти дозволені на середній шкалі Cb і Cr не для блоку 2x2, для рожевого і білого (вертикальний, горизонтальний) пікселів, тобто для 1x4, блоків 4x1 (схема 4: 1: 2), а також (схема 4: 1). Припустимо, що існують нові типи пророцтв для Cb і Cr, а також для практичних методів, які зупинилися.

Далі яскравості компонентів Y та відповідають за колір компоненти Cb і Cr розбиваються на блоки 8x8 пікселів. Кожен такий блок піддається дискретному косинусному перетворенню (ДКП). Отримані коефіцієнти ДКП квантуються (для Y, Cb і Cr в загальному випадку використовуються різні матриці квантування) і пакуються з використанням кодування серій і кодів Хаффмана. Стандарт JPEG допускає також використання значно більш ефективного арифметичного кодування, однак через патентні обмежень (патент на описаний в стандарті JPEG арифметичний QM-кодер належить IBM) на практиці воно використовується рідко. У популярну бібліотеку libjpeg останніх версій включена підтримка арифметичного кодування, але з переглядом стислих з використанням цього методу зображень можуть виникнути проблеми, оскільки багато програм перегляду не підтримують їх декодування.

Матриці, що використовуються для квантування коефіцієнтів ДКП, зберігаються в головній частині JPEG-файлу. Зазвичай вони будуються так, що високочастотні коефіцієнти піддаються більш сильному квантуванню, ніж низькочастотні. Це призводить до огрубіння дрібних деталей на зображенні. Чим вище ступінь стиснення, тим більше сильному квантуванню піддаються всі коефіцієнти.

При збереженні зображення в JPEG-файлі вказується параметр якості, що задається в деяких умовних одиницях, наприклад, від 1 до 100 або від 1 до 10. Більше число зазвичай

відповідає кращій якості (і більшого розміру стисненого файлу). Однак навіть при використанні найвищої якості (відповідного матриці квантування, що складається з одних тільки одиниць) відтворене зображення не буде в точності співпадати з вихідним, що пов'язано як з кінцевою точністю виконання ДКП, так і з необхідністю округлення значень Y, Cb, Cr і коефіцієнтів ДКП до найближчого цілого. Режим стиснення Lossless JPEG, який не використовує ДКП, забезпечує точний збіг відновленого і вихідного зображень, проте його мала ефективність (коефіцієнт стиснення рідко перевищує 2) і відсутність підтримки з боку розробників програмного забезпечення не сприяли популярності Lossless JPEG.

2.5 Висновки до другого розділу

В ході роботи над другим розділом були описані алгоритми та технології, на яких базується запропонований метод.

Детальний опис CDN надає інформацію про існуючі топології та опис Cloudflare – веб-сервісу, один з провідних сучасних мереж доставки контенту. Сервери цієї фірми розташовані в стратегічно обраних дата центрах по всьому світу, щоб прискорити доставку статичних файлів.

В кешуванні описується три основні механізми керування кешами та їх застосування в програмній реалізації.

Мінімізація та стиснення описують застосування утиліти GZip та веб-сервісу Minifiner.org. Також присутній опис методу стиснення JPEG.

Практичне застосування запропонованого методу удосконалення крос-платформної оптимізації веб-сайтів та визначення його переваг над існуючими рішеннями представлено в третьому розділі.

РОЗДІЛ 3
ПРАКТИЧНЕ ЗАСТОСУВАННЯ

Практичне застосування запропонованого методу оптимізації крос-платформної оптимізації веб-сайту проведене з використанням веб-сайту https://subdimensionsoft.eu/ – сайт візитка компанії розробників програмного забезпечення та ігор.

Апаратне забезпечення, що використане для проведення експерименту з практичного застосування методу, має наступні характеристики:

1) Операційна система – 64-розрядна, Windows 7.

2) Оперативна пам’ять – 4 Гб.

3) Процесор – AMD E1-11200 APU з Radeon HD Graphics (1,4 ГГц).

При тестуванні продуктивності на прикладі обраного сайту було застосовано сімейство інструментів PageSpeed Insights API (далі PSI). PSI повідомляє про ефективність сторінки як на мобільних, так і на настільних пристроях, а також надає пропозиції щодо покращення цієї сторінки.

PSI надає як лабораторні, так і польові дані про сторінку. Дані лабораторії корисні для налагодження проблем із продуктивністю, оскільки вони зібрані в контрольованому середовищі. Однак це може не відображати вузькі місця в реальному світі. Польові дані корисні для захоплення справжнього, реального досвіду користувачів, але мають більш обмежений набір показників.[25]

У верхній частині звіту PSI надає оцінку, яка підсумовує ефективність сторінки. Ця оцінка визначається запуском Lighthouse для збору та аналізу лабораторних даних про сторінку. Оцінка 90 або вище вважається швидкою, а від 50 до 90 вважається середньою. Нижче 50 вважається повільним.

Для достовірності отриманих даних перевіримо початкові дані сайту. Для цього перейдемо на сайт ресурсу PSI та введемо URL веб-сторінки та клікнути на кнопку Аналізувати (див. рис. 3.1).

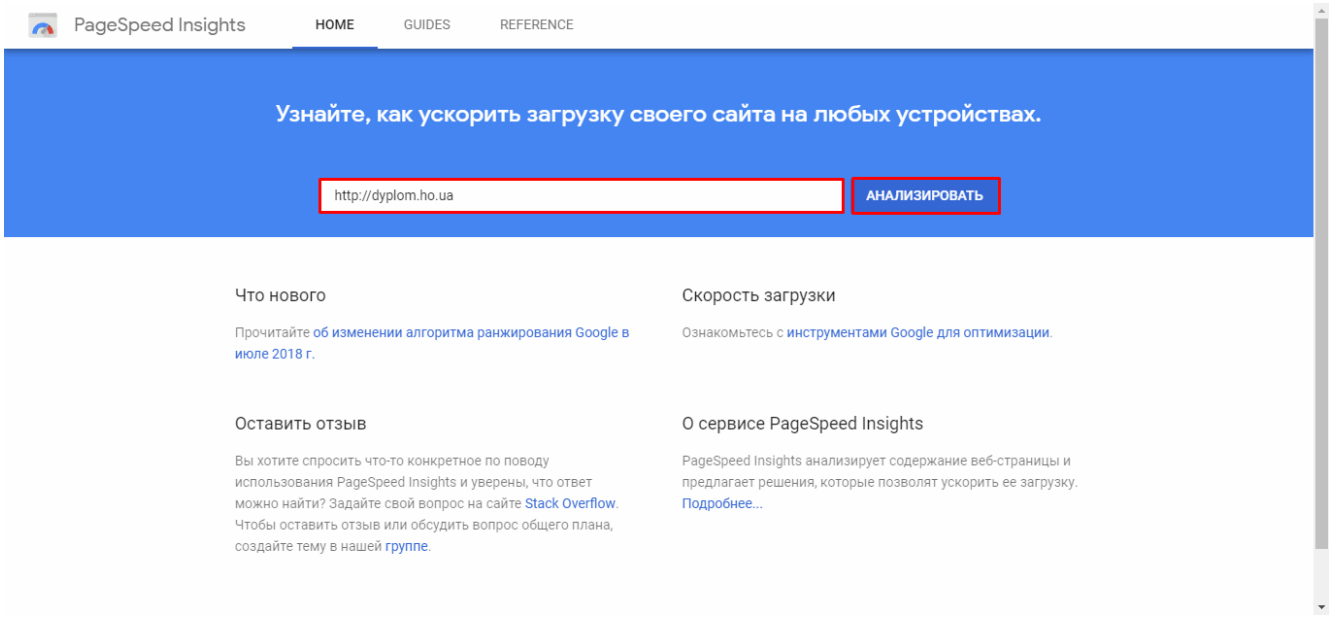


Рисунок 3.1 – Введення URL для аналізу веб-сторінки

Дослідження показало високі результати для мобільних пристроїв (див. рис. 3.2) та для комп’ютерів (див. рис. 3.3).

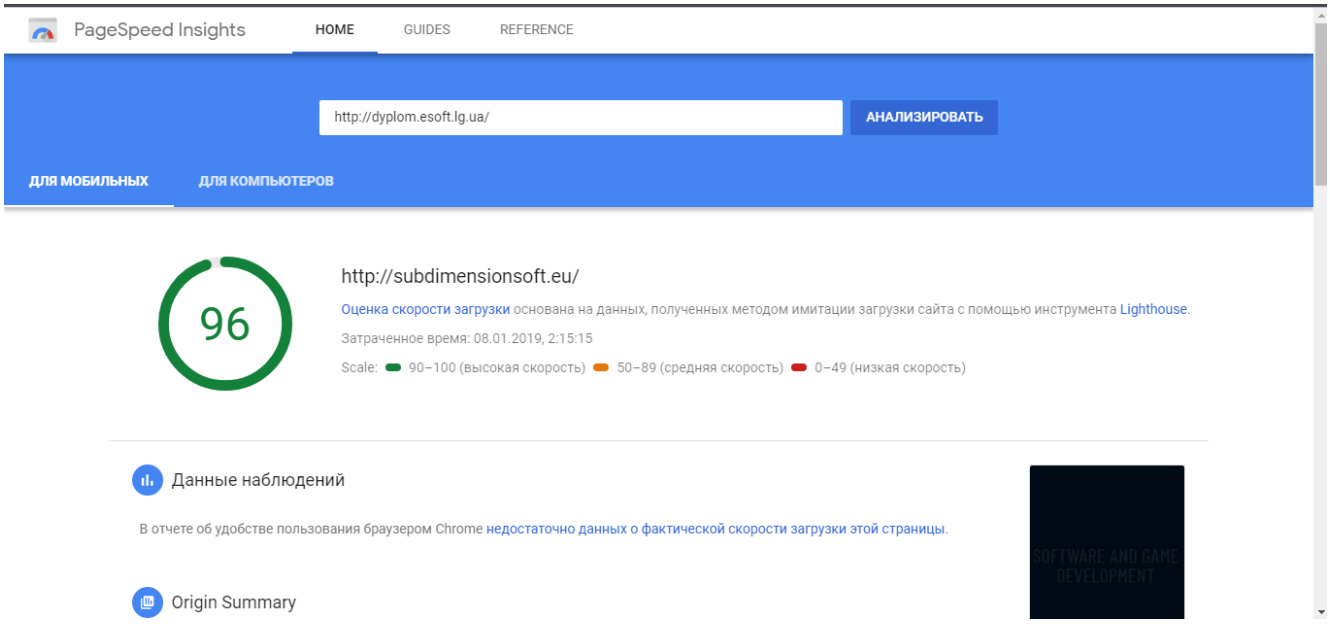


Рисунок 3.2 – Результат PSI аналізу для мобільних пристроїв

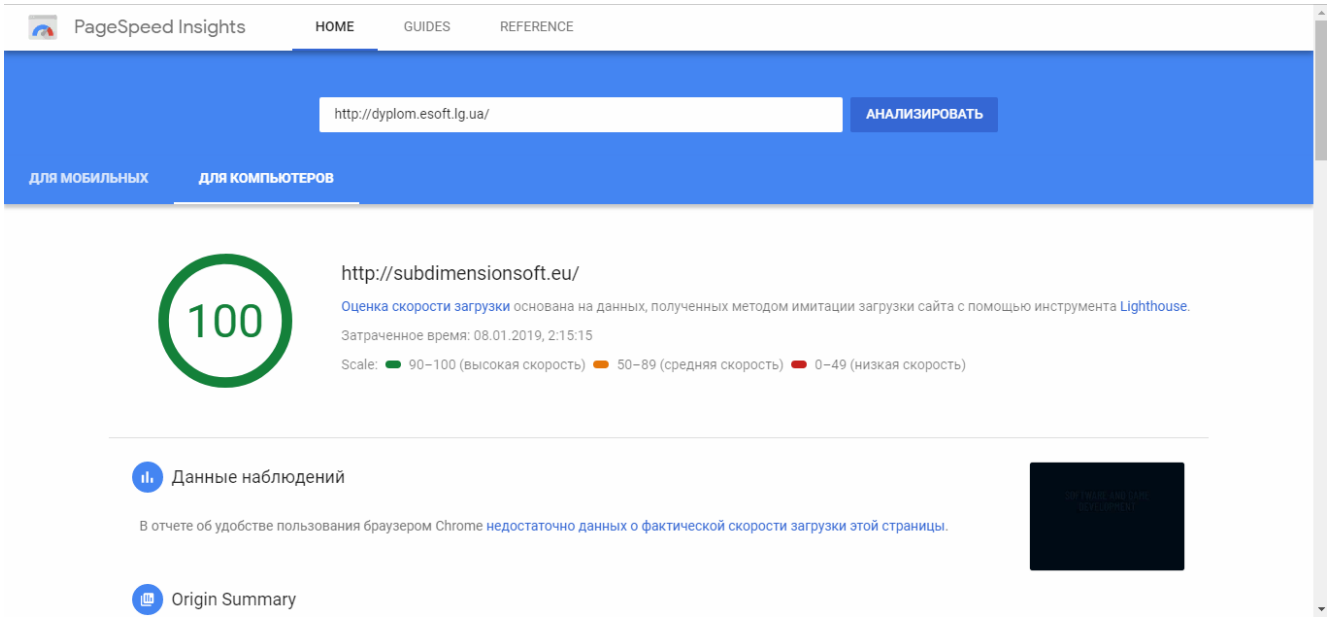


Рисунок 3.3 – Результат PSI аналізу для комп’ютерів

Окрім цього сервіс надає характеристики сайту, що формуються завдяки емульованої 3G-мережі (див. табл. 3.1)

Таблиця 3.1 – Початкові характеристики, отримані за допомогою PSI

Параметри, що перевіряються	Для мобільних пристроїв	Для стаціонарних пристроїв
Час завантаження першого контенту (секунд)	1,2	0,4
Індекс швидкості завантаження (секунд)	1,6	0,9
Час завантаження для взаємодії (секунд)	2,7	0,5
Час завантаження достатньої частини контенту (секунд)	1,8	0,5
Час закінчення роботи ЦП (секунд)	2,7	0,6
Приблизний час затримки при введенні (мілісекунд)	10	10

Для вирішення описаних проблем були використанні технології, що описані в розділі 2. Більш детально про їхнє застосування нижче.

3.1 Етап реалізації технології Content Delivery Network

Підключення CDN до сайту проведено за допомогою ресурсу Cloudflare. Cloudflare - американська компанія, що надає послуги CDN, розподілу DDoS, Інтернет-захисту сервісів і що знаходяться між користувачем і сайтом DNS серверів, що працюють як зворотній проксі для сайту [26].

Після реєстрації на офіційному сайті власний сайт додається в аккаунті (рис. 3.4). Форма для додавання сайту має поле введення адреси сайту та кнопку «Додати сайт». Ресурс працює лише з сайтами 1 рівня, тобто застосовувати субдомени заборонено.

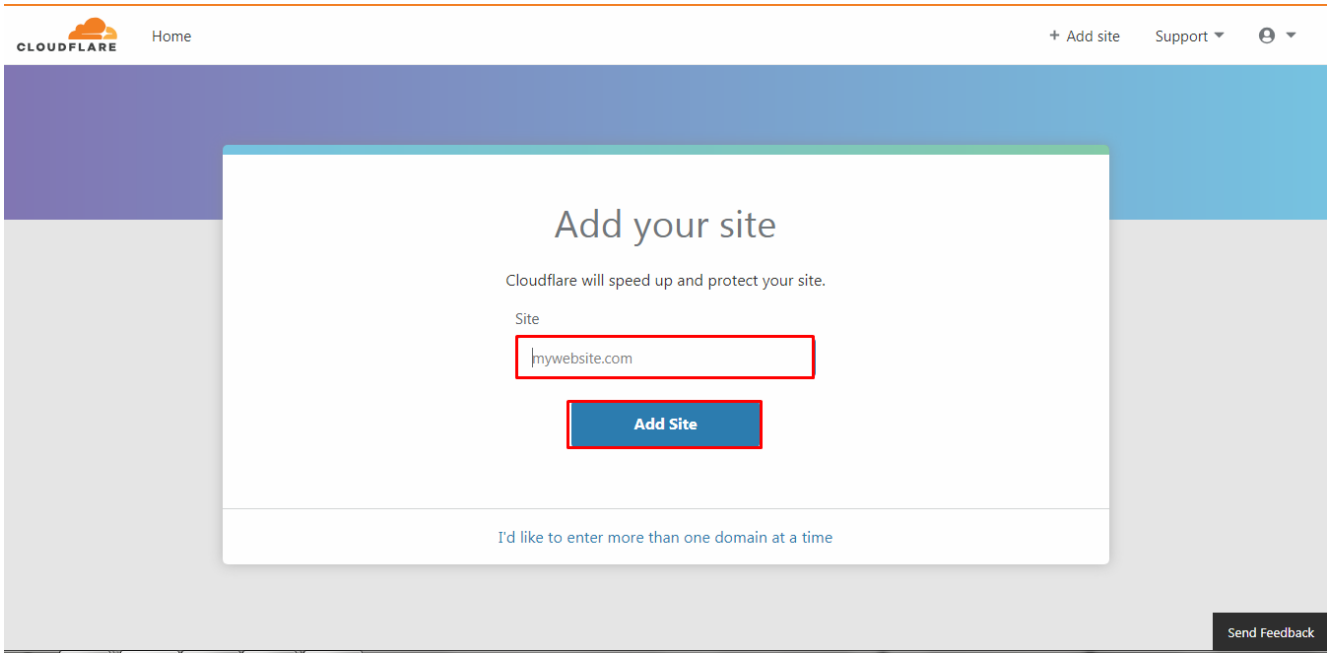


Рисунок 3.4 – Реєстрація сайту на ресурсі CloudFlare

Ресурс запитує існуючі записи DNS сайту, що надає можливість не записувати їх вручну (див. рис. 3.5).

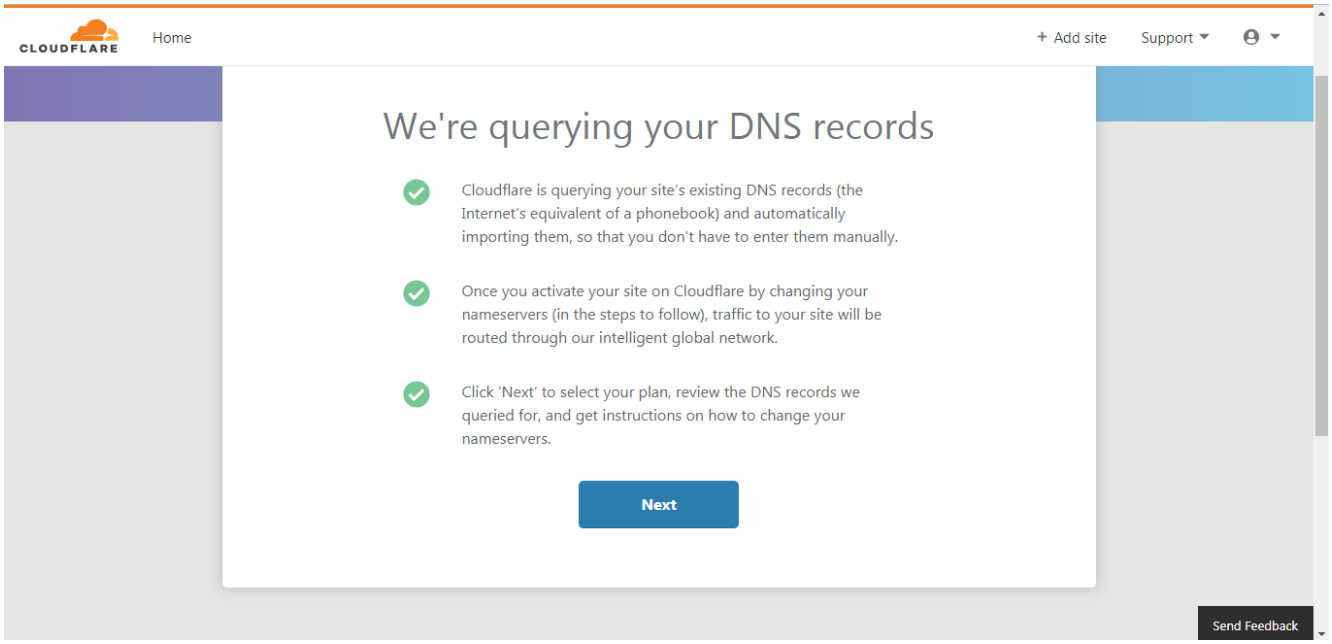


Рисунок 3.5 – DNS запити

Після цього Cloudflare надає посилання на нові сервери, де будуть знаходиться зображення та відеозаписи веб сайту (рис. 3.6). Зміна серверів необхідна для активації послуг ресурсу. Як тільки активується сайт на Cloudflare, трафік на сайт буде маршрутизований через інтелектуальну глобальну мережу Cloudflare.

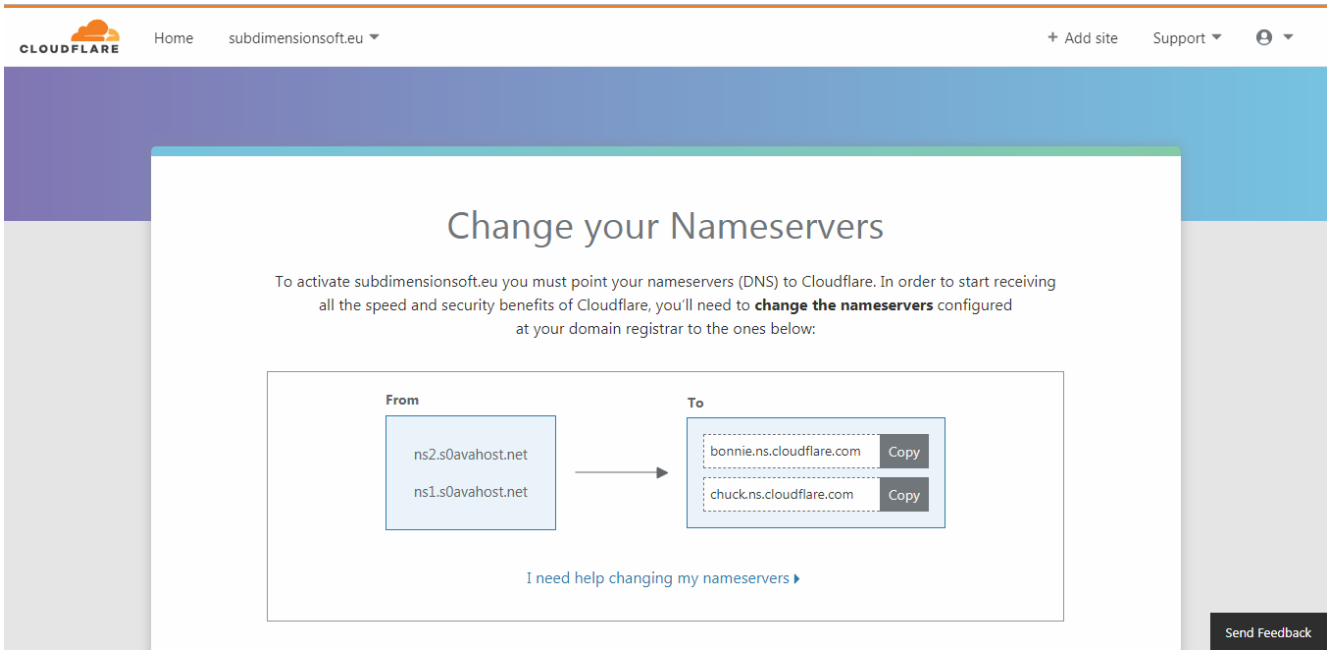


Рисунок 3.6 – Заміна DNS серверів на імена Cloudflare

Після налаштування імен серверів потрібно прописати правильні маршрути до файлів, які будуть знаходитися на CDN ресурсі. Для цього було записано код в файлі index.php наступного вигляду:

```
<!doctype html>
<html>
    <?php
    define('cdnURL',
'http://bonnie.ns.cloudflare.com.subdimensionsoft.eu/');
    ?>
<title>SubDimenisionSoft</title>
<head>
    <meta http-equiv='X-UA-Compatible' content='IE=Edge' />
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="keywords" content="SubDimenisionSoft, soft, game, games">
    <meta name="description" content="SubDimenisionSoft" />
    <meta name="author" content="admin" />
    <link rel="stylesheet" href="<?php echo cdnURL?>css/animate.css"
type="text/css">
    <link rel="stylesheet" href=" <?php echo cdnURL?> css/reset.css"
type="text/css">
    <link rel="stylesheet" href="<?php echo cdnURL?>css/style.css"
type="text/css">
    <link rel="stylesheet" href="<?php echo cdnURL?>css/media.css"
type="text/css">
```

Для простоти додавання повного маршруту використовуємо змінну cdnURL, в якій і прописуємо новий шлях до css, javascript та медіа файлів. За допомогою рядків коду `<?php echo cdnURL?>` прописується повний маршрут для кожного з файлів, що підключені до сайту. Повний лістинг сайту зображений в додатку А.

Після використання даних маніпуляцій було перенесено основні файли до ресурсу CloudFlare. Умови проведення експерименту по перевірці ефективності розробленого методу оптимізації мають на увазі найближче розташування серверу. При існуючих умовах різниця початкових даних та результатів, отриманих при проведенні експерименту, не є значною, оскільки технологія CDN базується на розміщенні контенту на серверах найближчих до користувачів.

Також, застосовувати цей метод оптимізації необхідно при великих навантаженнях на сервер великою кількістю користувачів, оскільки інакше на використання CDN будуть йти великі суми та відсутня вигода.

3.2 Етап реалізації технології кешування

Процес кешування проведений наступним чином. Правильним кешуванням вважається кешування засобами .htaccess. .htaccess - файл додаткової конфігурації веб-сервера Apache, а також подібних йому серверів. Дозволяє задавати велику кількість додаткових параметрів і дозволів для роботи веб-сервера в окремих каталогах (папках), таких як керований доступ до каталогів, перепризначення типів файлів і т.д., без зміни головного конфігураційного файлу.

.htaccess є подобою httpd.conf з тією різницею, що діє тільки на каталог, в якому розташовується, і на його дочірні каталоги. Можливість використання .htaccess в тому чи іншому каталозі вказується в httpd.conf (директива AllowOverride).

Файл .htaccess може бути розміщений в будь-якому каталозі. Директиви цього файлу діють на всі файли в поточному каталозі і у всіх його підкаталогах (якщо ці директиви не перевизначені директивами нижче лежачих файлів .htaccess). Для того щоб ці файли .htaccess можна було використовувати, необхідні відповідні налаштування головного конфігураційного файлу (значення директиви AllowOverride має бути встановлено All). Як правило, переважна більшість хостерів дозволяють використовувати свої файли .htaccess [27].

Для застосування методу кешування в файл . htaccess доданий наступний код:

```
AddHandler application/x-httpd-php .html
AddHandler cgi-script .pl .py .jsp .asp .htm .shtml .sh .cgi
AddType application/x-javascript .js
AddType text/css .css
AddType text/xml .xml
AddType application/octet-stream .doc .mov .avi .pdf .xls
# ForceType application/x-httpd-php
<ifModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/plain text/xml
application/xml application/xhtml+xml text/css text/javascript
application/javascript application/x-javascript
</ifModule>
<ifModule mod_headers.c>
    #кэшировать html и htm файлы на один день
    <FilesMatch "\.(html|htm)$">
        Header set Cache-Control "max-age=43200"
    </FilesMatch>
    #кэшировать css, javascript и текстовые файлы на одну неделю
    <FilesMatch "\.(js|css|txt)$">
        Header set Cache-Control "max-age=604800"
    </FilesMatch>
```

```
        #кэшировать флэш и изображения на месяц
        <FilesMatch "\.(flv|swf|ico|gif|jpg|jpeg|png)$">
            Header set Cache-Control "max-age=2592000"
        </FilesMatch>
        #отключить кэширование
        <FilesMatch "\.(pl|php|cgi|spl|scgi|fcgi)$">
            Header unset Cache-Control
        </FilesMatch>
    </IfModule>
    <ifModule mod_expires.c>
        ExpiresActive On
        #по умолчанию кеш в 5 секунд
        ExpiresDefault "access plus 5 seconds"
        #кэшировать флэш и изображения на месяц
        ExpiresByType image/x-icon "access plus 2592000 seconds"
        ExpiresByType image/jpeg "access plus 2592000 seconds"
        ExpiresByType image/png "access plus 2592000 seconds"
        ExpiresByType image/gif "access plus 2592000 seconds"
        ExpiresByType   application/x-shockwave-flash   "access   plus   2592000
seconds"
        #кэшировать css, javascript и текстовые файлы на одну неделю
        ExpiresByType text/css "access plus 604800 seconds"
        ExpiresByType text/javascript "access plus 604800 seconds"
        ExpiresByType application/javascript "access plus 604800 seconds"
        ExpiresByType application/x-javascript "access plus 604800 seconds"
        #кэшировать html и htm файлы на один день
        ExpiresByType text/html "access plus 43200 seconds"
        #кэшировать xml файлы на десять минут
        ExpiresByType application/xhtml+xml "access plus 600 seconds"
    </ifModule>
```

При використанні кешування на сайті повинно частково вирішити першу та повністю вирішити другу проблеми з сайтом. Після надання сайту кешувати файли перевіримо його за допомогою того ж PSI та перевірити результат кешування (див 3.7).

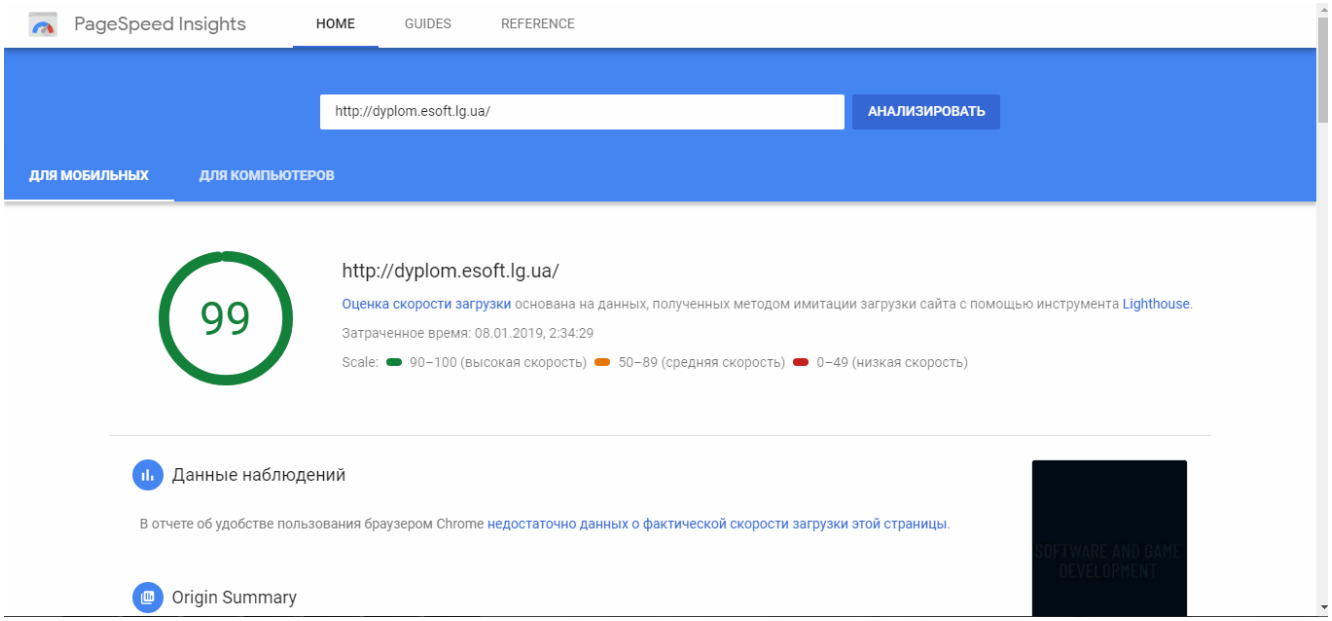


Рисунок 3.7 – Результат використання кешування засобами .htaccess

Як зображено на рисунку 3.7, в порівнянні з початковим тестом було виявлено підвищення оцінки швидкості завантаження сайту на 3%. При цьому, стартове завантаження сайту складає 7,85 секунд, а завантаження, що використовує кеш – 4, 98 секунд (див. табл. 3.2).

Таблиця 3.2 – Результат використання кешування

Завантаження сторінки	Час завантаження до застосування (секунд)	Час завантаження після застосування (секунд)
Перше	7,85	7,85
Наступні	5,34	4,98

Виходячи з результатів тестування, цей метод необхідно додати до ряду методів крос-платформної оптимізації веб-сайтів.

3.3 Етап реалізації технології мінімізації та оптимізації коду

Процес мінімізації та оптимізації є одним з найголовніших методів підвищення продуктивності та швидкості роботи сайту. Для підвищення швидкості завантаження

мінімізувані JavaScript та CSS за допомогою ресурсу Minify. Результати показали зменшення швидкості першого завантаження до 4,96 секунд.

Для досягнення найкращого результату використані утиліта стиснення GZip описані в розділі. Для реалізації даної утиліти необхідно прописати в файл .htaccess наступні правила:

```
<IfModule mod_deflate.c>
  AddOutputFilterByType DEFLATE text/html
  AddOutputFilterByType DEFLATE text/css
  AddOutputFilterByType DEFLATE text/javascript
  AddOutputFilterByType DEFLATE text/xml
  AddOutputFilterByType DEFLATE text/plain
  AddOutputFilterByType DEFLATE image/x-icon
  AddOutputFilterByType DEFLATE image/svg+xml
  AddOutputFilterByType DEFLATE application/rss+xml
  AddOutputFilterByType DEFLATE application/javascript
  AddOutputFilterByType DEFLATE application/x-javascript
  AddOutputFilterByType DEFLATE application/xml
  AddOutputFilterByType DEFLATE application/xhtml+xml
  AddOutputFilterByType DEFLATE application/x-font
  AddOutputFilterByType DEFLATE application/x-font-truetype
  AddOutputFilterByType DEFLATE application/x-font-ttf
  AddOutputFilterByType DEFLATE application/x-font-otf
  AddOutputFilterByType DEFLATE application/x-font-opentype
  AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
  AddOutputFilterByType DEFLATE font/ttf
  AddOutputFilterByType DEFLATE font/otf
  AddOutputFilterByType DEFLATE font/opentype
# For Older Browsers Which Can't Handle Compression
  BrowserMatch ^Mozilla/4 gzip-only-text/html
  BrowserMatch ^Mozilla/4\.0[678] no-gzip
  BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
</IfModule>
```

Після додання цих правил оцінка швидкості завантаження сайту не підвищилась, однак якщо порівняти з аналізом, який зображений на рисунку 3.7, то можна побачити, що ряд проблем (див. рис. 3.8) зі швидкістю завантаження сайту були вирішені (див. рис. 3.9).

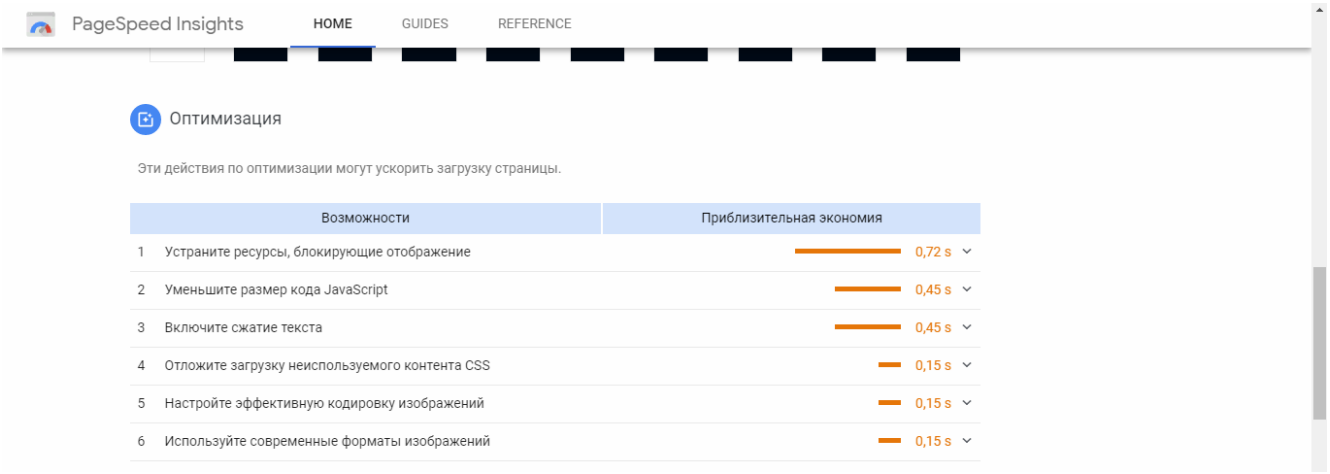


Рисунок 3.8 – Аналіз сайту до застосування GZip

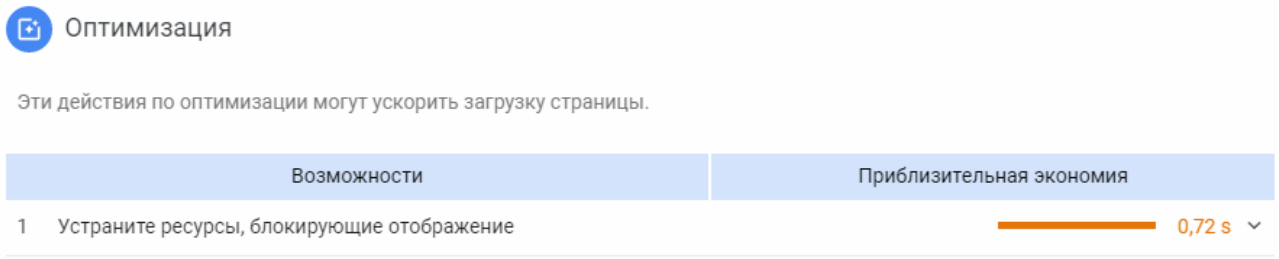


Рисунок 3.9 – Аналіз сайту після застосування GZip

При використанні цієї утиліти збільшилась швидкість початкового завантаження – до 4,46 секунд та без використання кешування.

Комбінуючи кешування та утиліту GZip було отримано, що швидкість початкового завантаження сторінки складає 4, 36 секунд, а наступних – 4,05 секунд, що демонструє правильність застосування обох методів для крос-платформної оптимізації веб-сайтів.

Таблиця 3.3 – Результат використання GZip та мінімізації

Завантаження сторінки	Час завантаження до застосування (секунд)	Час завантаження після застосування (секунд)
Перше	7,85	4,36
Наступні	5,34	4,05

При об’єднанні цих способів та результату мінімізації веб-ресурсом Minify був отриманий результат при наступних завантаженнях 3,68 секунд, що говорить про необхідність використання даного веб-ресурсу.

3.4 Етап реалізації технології стискання зображень

Останнім, та не менш важливим є стискання зображень та їх конвертація в сучасні формати. Перш за все необхідно застосувати один з наведених ресурсів для перетворення зображень із початкових форматів в формат WebP.

Для цього завантажимо всі зображення на веб-ресурс IMGonline окремо, виберемо формат зображення на виході, після чого необхідно обрати параметри стискання для формату WebP та обрати якість, запропоновану PSI – 85% (див. рис. 3.10). Після конвертації, IMGonline відобразить сторінку з розміром файлу та можливістю його закачування (див. рис. 3.11).

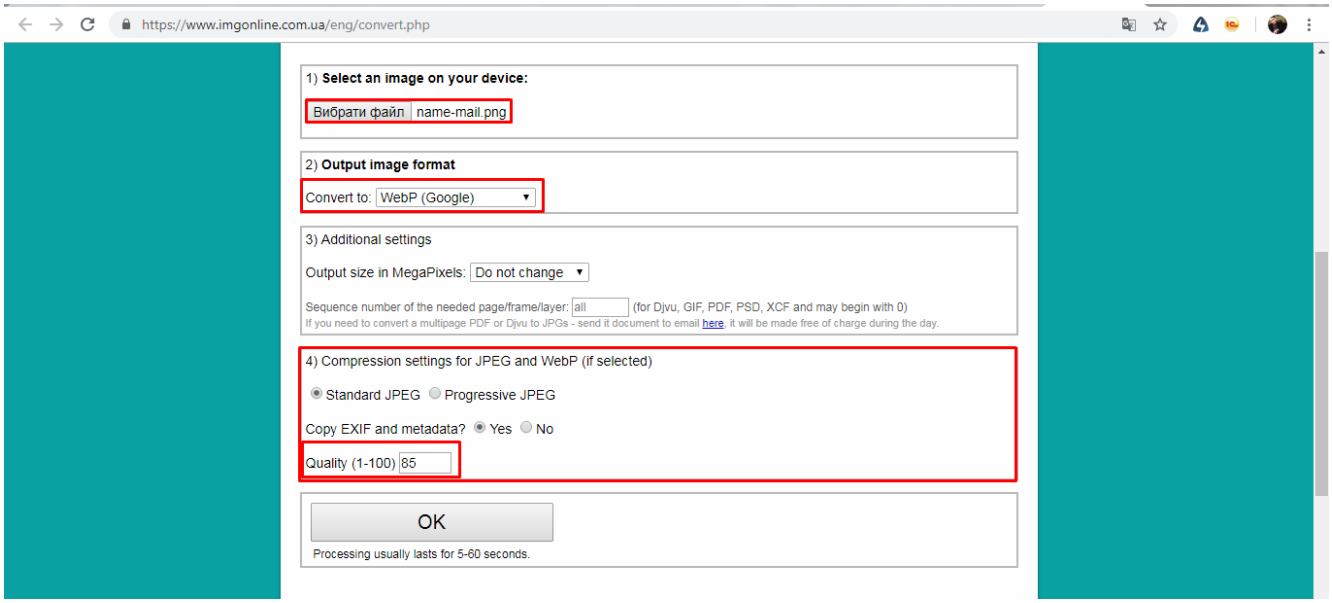


Рисунок 3.10 – Підготовка до конвертації

Рисунок 3.11 – Результат конвертації

Після конвертації зображень необхідно замінити існуючі зображення перетвореними та прописати нові шляхи. Наприклад, `img/bg1.jpg` необхідно змінити на `img/bg1_zip.webp`.

Після додавання змінених зображень швидкість першого завантаження знизилась до 5,04 секунд та 4,69 секунд наступних завантажень без використання попередніх методів (див. табл. 3.4).

Таблиця 3.4 – Результат використання конвертації

Завантаження сторінки	Час завантаження до застосування (секунд)	Час завантаження після застосування (секунд)
Перше	7,85	5,04
Наступні	5,34	4,69

При комбінуванні кешування та змінення формату зображення швидкість першого завантаження збільшилась до 4,69 секунд та наступні завантаження – до 4,16 секунд.

При комбінуванні мінімізації, утиліти GZip та змінення формату зображення показали підвищення швидкості першого завантаження до 4,57 секунд та 3,81 секунд – наступних.

Комбінування кешування, мінімізації, використання утиліти GZip та зміни формату зображення привели до підвищення першого завантаження за 4 секунди та при наступних

завантаженнях до 3,2 секунд. Результати об’єднання показали використання цих методів значно оптимізує веб-сайт, що також демонструє сервіс PageSpeed Insights (див. рис. 3.12, рис. 3.13).

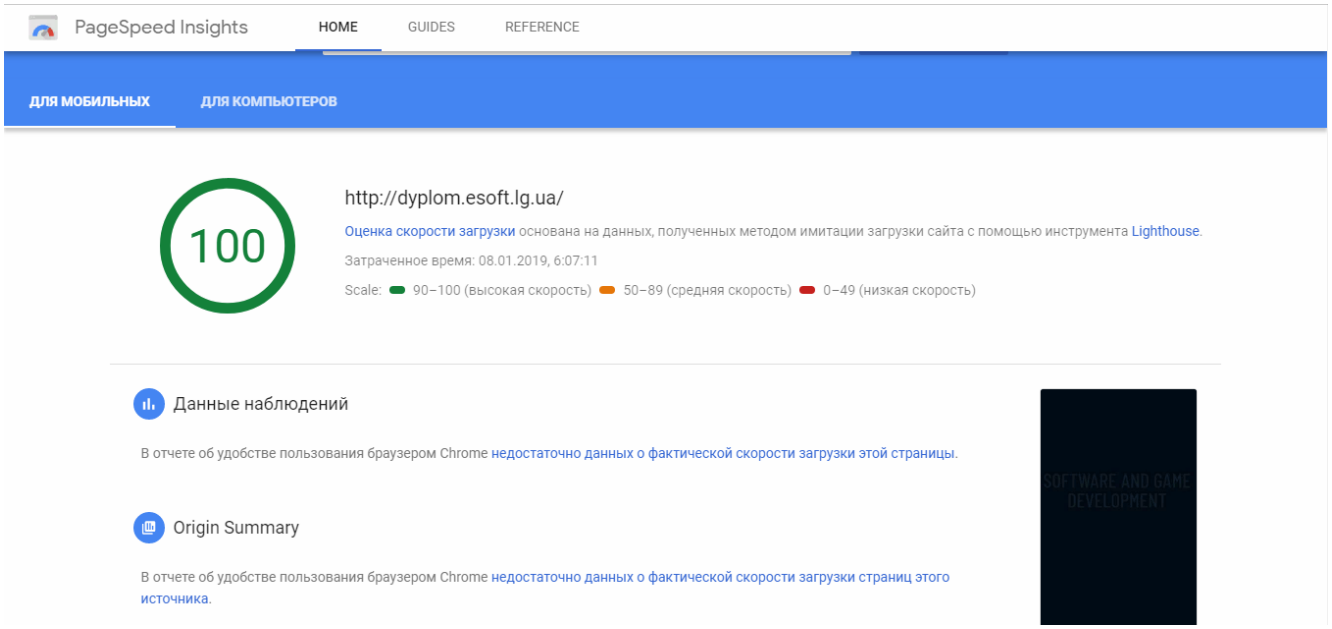


Рисунок 3.12 – Оцінка швидкості завантаження для мобільних пристроїв після комбінування методів

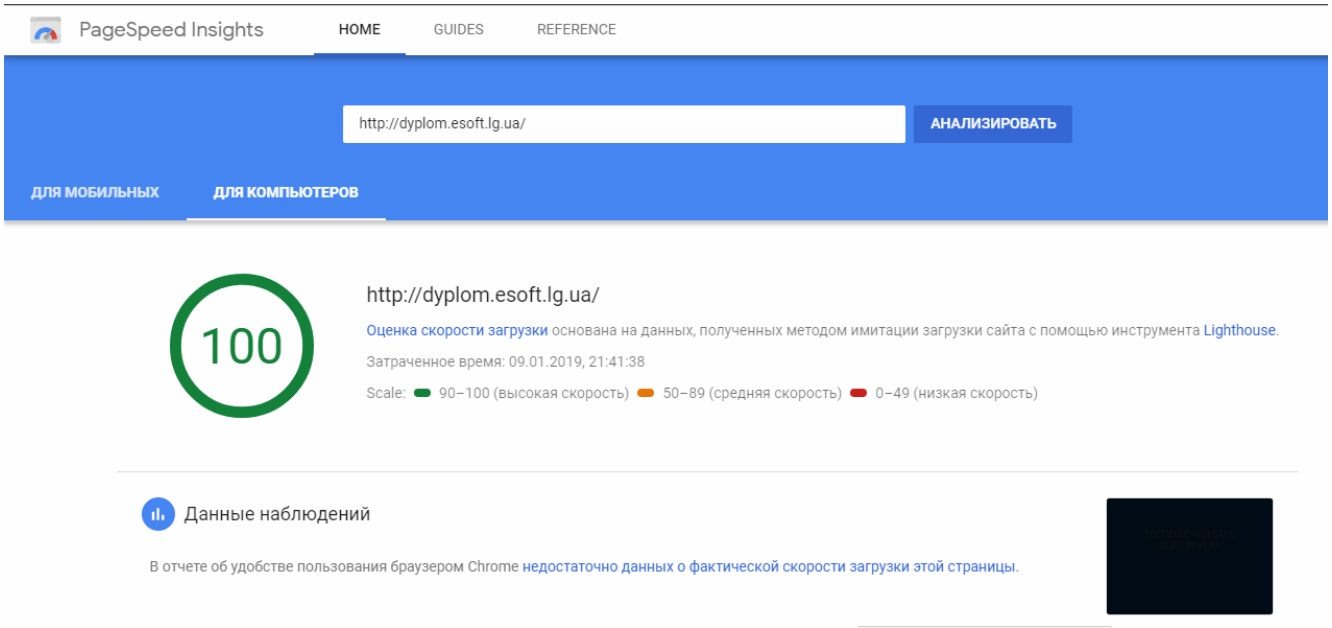


Рисунок 3.13 – Оцінка швидкості завантаження для стаціонарних пристроїв після комбінування методів

Також в результаті дослідження були нові характеристики імітації завантаження сторінки (див. табл. 3.5).

Таблиця 3.5 –Характеристики, отримані за допомогою PSI після запропонованого методу

Параметри, що перевіряються	Для мобільних пристроїв	Для стаціонарних пристроїв
Час завантаження першого контенту (секунд)	1,2	0,4
Індекс швидкості завантаження (секунд)	1,6	0,9
Час завантаження для взаємодії (секунд)	2,7	0,5
Час завантаження достатньої частини контенту (секунд)	1,8	0,5
Час закінчення роботи ЦП (секунд)	2,7	0,6
Приблизний час затримки при введенні (мілісекунд)	10	10

3.5 Висновки до третього розділу

В третьому розділі представлене практичне використання та визначена переваша запропонованого методу удосконалення крос-платформної оптимізації веб-сайтів.

В результаті проведення експерименту, при використанні CDN було виявлено, що його використання в даній роботі не є обов’язковим, оскільки відсутні помітні зміни при його застосуванні. Також було виявлено, що оскільки мережа доставки контенту на платній основі, її застосування при невеликих проектах є більш збитковою, аніж прибутковою. В процесі реалізації було вирішено не використовувати Content Delivery Network в даній роботі.

Кешування мало одне з найважливіших місць в даній роботі, оскільки його використання в сукупності з іншими методами привело до зменшення часу завантаження сторінок.

Великий вплив на швидкість надали мінімізація коду та використання утиліти GZip, оскільки їх використання підвищило швидкість не тільки наступних завантажень, яке надавав і метод кешування, а й зменшення часу при першому завантаженні.

Незначний, але важливий вплив на оптимізацію веб-сайту в цілому надала конвертація зображень та їх стиснення.

В результаті використання запропонованого методу удосконалення крос-платформної оптимізації з використанням кешування: свіжість, валідація та інвалідація, мініфікації та стиснення зображень отримані наступні результати, представлені в табл. 3.6.

Таблиця 3.6 – Результати оптимізації

Технології	До застосування методів		Запропонований метод	
	Перше завантаження	Наступні завантаження	Перше завантаження	Наступні завантаження
CDN	7,85	5,34	-	-
Кешування			7,85	4,98
Мініфікація			4,36	4,05
Стиснення зображень			5,04	4,69

Порівняльний аналіз в табл. 3.6 показує переваги розробленого методу удосконалення крос-платформної оптимізації веб-сайту. В результаті використання запропонованого методу оптимізовано завантаження веб сайту та збільшено швидкість першого завантаження з 6,22 секунд до 4 секунд та з 5,22 секунд до 3,2 секунд на етапі – при наступних завантаженнях.

Таким чином, вирішено поставлене завдання - підвищення продуктивності веб-сайту.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики. Завданням даної магістерської роботи було дослідити та удосконалити метод крос-платформної оптимізації сайтів. Так як в процесі проектування використовувалося електронне обладнання, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для робочого місця, з використанням персонального комп'ютера на якому розроблена інтелектуальна охоронна система.

4.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. У законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

При роботі з обчислювальною технікою змінюються фізичні і хімічні фактори навколишнього середовища: виникає статична електрика, електромагнітне випромінювання, змінюється температура і вологість, рівень вміст кисню і озону в повітрі. Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, які використовуються для обробки приміщень та обладнання. Неправильна організація робочого місця сприяє загальному і локальній напрузі м'язів шиї, тулуба, верхніх кінцівок, викривлення хребта і розвитку остеохондрозу.

4.1.2 Організаційно-технічні заходи з безпеки праці

У організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки

знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 [28].

Також впроваджені організаційні заходи з пожежної безпеки - навчання і перевірку знань відповідно до вимог Типового положення про інструктажі в установах та організаціях України, затвердженого наказом Міністерства України з питань надзвичайних ситуацій.

Обов'язковими вимогами враховане наступне:

– не слід допускати до роботи осіб, що в установленому порядку не пройшли навчання, інструктаж та перевірку знань з охорони праці, пожежної безпеки та цих Правил.

– ознайомлення з правилами безпеки праці, одержання відповідних інструктажів засвідчується у журналі інструктажів.

– перед допуском до самостійної роботи кожен працівник проходить навчання з питань охорони праці:

1) вступного, який проводять працівники служби охорони праці об'єкта господарювання з усіма працівниками, яких приймають на роботу незалежно від їхньої освіти та стажу роботи за програмою, в якій подають загальні питання охорони праці із врахуванням її особливостей на об'єкті господарювання;

2) первинного, який проводять керівники структурних підрозділів на місці праці з кожним працівником до початку їхньої роботи на цьому робочому місці.

– обов'язкові організаційні заходи перед початком, під час і після завершення роботи повинні включати перевірку (візуально) наявності і справності електрообладнання та його заземлення, а під час виконання роботи вимогу «не залишати без нагляду обладнання, яке працює». Після закінчення роботи - вимагається прибирання робочого місця, відключення всіх електроприладів від електромережі.

4.2 Аналіз стану умов праці

Робота над створенням програмного засобу логічного виводу інформаційної підтримки прийняття рішень з управління небезпечними об'єктами у критичних ситуаціях буде проходити у приміщенні дев'ятиповерхового будинку на 1-ому поверсі. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

4.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені у табл. 4.1. Згідно з [30] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 м^2, а об'єм — не менше 20 м^3.

Таблиця 4.1 – Розміри приміщення

Найменування	Значення
Довжина, м	4
Ширина, м	4
Висота, м	3
Площа, м^2	16
Об'єм, м^3	48

Отже, дане приміщення цілком відповідає зазначеним нормам. Для зручності спільної роботи з іншими працівниками у кімнаті є диван, стільці та стіл. Також робочий процес пов'язаний з багатьма документами для чого приміщення облаштоване шафою. Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення. Для забезпечення потрібного рівного освітленості кімната має вікно та систему загального рівномірного освітлення, що встановлена на стелі.

4.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця (табл. 4.2) нормативним основні вимоги до організації робочого місця за [4] і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Робочий стіл на досліджуваному місці містить достатньо простору для ніг. Крісло, що використовується в якості робочого сидіння, є підйомно-поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки, також воно м'яке і виконане з екологічної шкіри, що дає можливість працювати у комфорті. Екран монітору знаходиться на відстані 750 мм, клавіатура має можливість регулювання кута нахилу 5-15°. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам.

Приміщення знаходиться на першому поверсі дев'ятиповерхової будівлі і має об'єм 48 м^3, площу – 16 м^2. У цьому кабінеті обладнане одне місце праці, укомплектоване ПК.

Таблиця 4.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	740	не менше 600
Ширина простору для ніг, мм	690	не менше 500
Глибина простору для ніг, мм	660	не менше 650
Висота поверхні сидіння, мм	420	400 ÷500
Ширина сидіння, мм	410	не менше 400
Глибина сидіння, мм	500	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	750	700 ÷800

Температура в приміщенні протягом року коливається у межах 20–22°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум у приміщені знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

Розміщення вікон забезпечує природне освітлення з коефіцієнтом природного освітлення не менше 1,5%, а загальне штучне освітлення, яке здійснюється за допомогою восьми люмінесцентних ламп, забезпечує рівень освітленості не менше 200 Лк.

У приміщені є електрична мережа з напругою 220 В, яка створює небезпеку ураження електричним струмом. ПК може бути джерелом електромагнітних випромінювань, аерозолів та шкідливих речовин (часток тонеру, оксидів нітрогену та озону).

За ступенем пожежної безпеки приміщення належить до категорії В.

У приміщені робиться вологе прибирання та провітрювання.

4.2.3 Навантаження та напруженість процесу праці

Під час виконання магістерської роботи: за фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до ІІ – ІІІ ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Під час виконання робіт використовується ПК, що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м'язово скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені в [31].

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни встановлено регламентовану перерву тривалістю 10 хв через кожну годину роботи.

4.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Робота, пов'язана з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконується із забезпеченням виконання [32], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП.

Основними робочими характеристиками персонального комп'ютера є наступні:

- робоча напруга U = +220В ± 5%;
- робочий струм I = 2А;
- споживана потужність P = 350 Вт.

Робочі місця відповідають вимогам Державних санітарних правил і нормам роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [31].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мікрокліматичні умови, освітлення, електромагнітні випромінювання, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3).

Таблиця 4.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
фізичні			
підвищена температура поверхонь обладнання	експлуатація ЕОМ	2	[30]
підвищена або знижена вологість повітря	-	2	[30]
підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-	4	[33], [34]
підвищена напруженість електричного поля	-	2	[35]
недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[36]
недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[37]
підвищена яскравість світла	порушення умов праці (організації місця праці- налагодження моніторів)	1	[37]
понижена контрастність	-	1	[37]
психофізіологічні:			
нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[32], [37]
фізичні (статичне – сидіння)	порушення умов праці (організації місця праці- сидіння користувача,) та організації робочого часу - безперервна робота)	2	[32], [37]

4.3.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ та вентиляції. Небезпека загоряння пов'язана з особливістю комп'ютерів - із значною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах. При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

Для відводу теплоти від ЕОМ діє потужна система кондиціонування. Тому кисень, як окиснювач процесів горіння, є в будь-якій точці приміщень ВЦ.

4.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК виконана як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника.

4.4 Гігієнічні вимоги до параметрів виробничого середовища

4.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря.

У даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, тому для нього відповідає категорія робіт Іа. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [38] і наведені в табл. 4.4:

Таблиця 4.4 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення та припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [30].

Рівні позитивних і негативних іонів у повітрі мають відповідати [30]. Для забезпечення оптимальних параметрів мікроклімату в приміщенні проводяться перерви у роботі співробітників, з метою його провітрювання.

4.4.2 Освітлення

Освітленість приміщення має велике значення при роботі на ПЕОМ. Вона багато у чому визначається колірною і мережевий обстановкою. Для зменшеного поглинання світла стеля і стіни вище панелей (1,5-1,7м). Для забарвлення стіни панелей віддана перевага світлим фарбам.

Основний потік природного світла - зліва.

У проекті, що розробляється, використовувалося суміщене освітлення. У світлий час доби використовувалося природне освітлення приміщення через віконні отвори, в решту часу використовувалося штучне освітлення. Штучне освітлення створюється єнергозберігаючими лампами.

Джерелом природного освітлення є сонячне світло. Регулярно проводиться контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для приміщення у світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2 + b^2} \cdot S_b \tag{4.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$$S_n = a \cdot b = 16 \text{ м}^2$$

$$S_{вік} = 1/8 \cdot 25 = 28{,}8 \text{ м}^2$$

Приймаємо 2 вікна площею S=3 м^2 кожне.

Світильники загального освітлення розташовуються над робочими поверхнями у рівномірно-прямокутному порядку. Для організації освітлення у темний час доби передбачається обладнати приміщення, довжина якого складає 4 м, ширина 4 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожні.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників *n* виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \tag{4.2}$$

де *E* – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 16 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо n=1,2.

Приймаємо освітлювальну установку, яка складається з одного світильника, який складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4 Вентилювання

Обмін повітря здійснюється при наскрізному провітрюванні.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

Заходи безпеки під час експлуатації персонального комп'ютера передбачають:

– правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

– експлуатацію сертифікованого обладнання;

– дотримання заходів електробезпеки;

– забезпечення оптимальних параметрів мікроклімату;

– забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

– облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря (об'єм приміщення 16 м3, тому потрібно подати не менш як 30 м3/год повітря).

– Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

– Загальний опір захисного заземлення визначається за формулою:

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п}, \tag{4.3}$$

де Rз - опір заземлення, якими когут бать труби, опори, кути і т.п., Ом;

Rш - опір опори, яке з'єднує заземлювачі, Ом;

п - кількість заземлювачів;

ηз - коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; ηз = 0,7;

ηш - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; ηш = 0,5.

Опір заземлення визначається за формулою:

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(\ln \frac{2 \cdot l}{d} + \frac{1}{2} \ln \frac{4 \cdot t + l}{4 \cdot t - l} \right), \tag{4.4}$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м;

для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 2 м.

Тоді опір заземлення дорівнює 110 Ом.

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_{ш} = \frac{\rho}{2\pi \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot t^1}, \quad (4.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: $П_{буд.} = 42 \cdot 2 + 38 \cdot 2 = 160$ м; L = 160 м;

b - ширина смуги, м; b = 0,03 м;

t_1 - глибина заземлення від рівня землі, м; $t_1 = 0,5$ м.

Тоді опір смуги, що з'єднує заземлювачі дорівнює 5,99 Ом

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з}, \quad (4.6)$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Тоді кількість заземлювачів захисного заземлення дорівнює 79.

Визначаємо, що загальний опір захисного заземлення дорівнює 1,7 Ом

Висновок: дане захисне заземлення забезпечує електробезпеку будівлі, так як виконується умова: $R_{ззп} < 4$ Ом.

У приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків та усуненню пожежі.

4.6 Охорона навколишнього природного середовища

4.6.1 Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: «Аналіз та розробка програмних засобів інформаційної підтримки прийняття рішень щодо управління небезпечними об'єктами в критичних ситуаціях» в процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом у процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на знешкодження, утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

У процесі діяльності виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- батарейки та акумулятори (малі) - ІІІ клас небезпеки;
- макулатура - IV клас небезпеки;
- побутові відходи - IV клас небезпеки;
- відпрацьовані люмінесцентні лампи - І клас небезпеки.

4.6.2 Визначення впливу та заходів щодо поводження з відходами ІТ галузі

З метою визначення та прогнозування впливу відходів на навколишне середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів. Відомості про місце утворення та місце розташування відходів зазначаються на «План схемі місці розміщення відходів організації / виробництва» та наводяться у таблиці 4.5.

Таблиця 4.5 - Відомості про місце утворення та місце розташування відходів

№ з/п	Код та найменування відходів за ДК - 005-96	Технологічний процес або виробництво, де утворюються відходи/клас небезпеки	Місце розташування відходу, тара та її кількість, місткість, розміри у разі наявності майданчиків розташування відходів необхідно зазначити тип покриття та наявність даху)	№ на схемі (додається масштабна схема місць розміщення відходів)
11	7710.3.1.26 Лампи люмінесцентні	1	буд. 6А, у приміщені S=16м^2, в кількість 2 од.	6А01-ТХ
22	7720.3.1.01 Відходи комунальні (міські) змішані, у т.ч. сміття з урн (Побутові відходи)	4	зовнішній майданчик зберігання побутових відходів біля буд. 6А S=8м² V= 1,0м³- 3од.	6А01-ТХ
33	7710.3.1.01 Макулатура паперова та картонна (Макулатура)		буд. 6А S =16,0 м.^{2}	6А01-ТХ
44	Батарейки та акумулятори (малі)	3	буд. 6А V=0,0005 м^3	6А01-ТХ

4.7 Висновок до четвертого розділу

У результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Було наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері. Визначені основні екологічні аспекти впливу на навколишнє природнє середовище та зазначені заходи щодо поводження з ними.

ВИСНОВКИ

Метою дипломної роботи визначено дослідження методів та технологій крос-платформної оптимізації сайтів та розробка власного методу, що забезпечує оптимізацію незалежно від платформи, що використовується, на якій розробляється та працює сайт. Метод повинен забезпечувати підвищення продуктивності та швидкості роботи сайту, працювати без збоїв в режимі реального часу.

В ході дослідницької частини роботи були отримані результати, за якими можна зробити наступні висновки:

1. Досліджено відомі технології оптимізації сайту, визначені кращі з існуючих технологій для реалізації розроблюваного методу оптимізації.

2. Визначена проблема оптимізації веб-сайту, яка полягає у відсутності універсального методу оптимізації.

3. Встановлено, що основним напрямком роботи крос-платформної оптимізації є створення та реалізація універсального методу удосконаленої крос-платформної оптимізації сайту.

4. Розроблено метод удосконаленої крос-платформної оптимізації сайту з використанням визначених технологій сучасної оптимізації.

В ході практичної частини роботи були отримані наступні результати:

1. Проведено експеримент з практичного застосування розробленого методу удосконалення крос-платформної оптимізації сайтів.

2. Отримані результати тестування методу, що демонструють високі показники збільшення швидкості завантаження майже на 65%.

Також, здійснений аналіз потенційно небезпечних і шкідливих виробничих факторів проектованого обладнання та наведено рекомендації щодо пожежної безпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1. DIGITAL IN 2018: WORLD'S INTERNET USERS PASS THE 4 BILLION MARK [Електронний ресурс] - Режим доступу: https://wearesocial.com/blog/2018/01/global-digital-report-2018 – 24.10.2018.

2. Статистика интернет-аудитории Украины и используемых устройств [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://seoukraine.com.ua/statistika-internet-auditorii-ukrainy-i-ispolzuemyh-ustroystv/ - 2.12.2018.

3. Кроссплатформенность [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Кроссплатформенность - 02.12.2018.

4. Ron Kohavi, Seven Rules of Thumb for Web Site Experimenters, Ron Kohavi, Alex Deng, Roger Longbotham, Ya Xu.

5. Побежимова М. П. Инструменты / подходы / способы оптимизации веб-приложений: на примере .net и Android / М. П. Побежимова, Е. И. Казимирова, М. В. Стержанов //Инновации в науке / Сб. ст. по материалам XLIX междунар. науч.-практ. конф. № 9 (46). Новосибирск: Изд. АНС «СибАК», 2015. С. 29-36.

6. Jiang Zhu, Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture, Jiang Zhu, Douglas S. Chan, Mythili Suryanarayana Prabhu та ін., 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering – 2013, С. 320-323.

7. Fog computing [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Fog_computing.

8. Сайт [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Сайт - 02.12.2018.

9. Taleb M. Pattern-Oriented Design Composition and Mapping for Cross-Platform Web Applications [Електронний ресурс] / M. Taleb, H. Javahery, A. Seffah. – 2006. – Режим доступу до ресурсу: https://www.researchgate.net/publication/228656223_Pattern-oriented_design_composition_and_mapping_for_cross-platform_Web_applications – 26.10.2018.

10. US 2009/0030859 A1, Method and apparatus for real-time website optimization, Francois Buchs, Zijad F Aganovic. – 11/880,823; filed 24.07.2007; publ. day 29.01.2009.

11. Muliono, Rizki. (2016). WEBSITE OPTIMIZATION CMS BASED ON GOOGLE PEGESPEED. Computer Engineering, Science and System Journal. 1. 32-35. – Режим доступу – https://www.researchgate.net/publication/315988500_WEBSITE_OPTIMIZATION_CMS_BASED_ON_GOOGLE_PEGESPEED. - 20.12.2018

12. Siroker, Dan. Systems and methods for website optimization. Siroker, Dan & Koomen, Pete & Kim, Elliot & Siroker, Eric. -2014. Режим доступу до ресурсу: https://www.researchgate.net/publication/302780425_Systems_and_methods_for_website_optimization

13. Content delivery network [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Content_delivery_network.

14. Majumdar S., Addressing Click Fraud in Content Delivery Systems., Majumdar Saugat & Kulkarni, Dhananjay & Ravishankar, Chinya – 2007 – 240 - 248. 10.1109/INFCOM.2007.36.

15. Cache(computing) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Cache_(computing).

16. Web cache [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Web_cache.

17. Web caching [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/caching/web-caching/.

18. Оптимизация кода [Електронний ресурс] – Режим доступу до ресурсу: https://www.viva64.com/ru/t/0084/.

19. Сжатие изображений [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Сжатие_изображений.

20. Iloveimg [Електронний ресурс] – Режим доступу до ресурсу: https://www.iloveimg.com/ru/compress-image.

21. IMGonline [Електронний ресурс] – Режим доступу до ресурсу: https://www.imgonline.com.ua/compress-image.php.

22. JPG Zipper [Електронний ресурс] – Режим доступу до ресурсу: http://zip.yutex.ru/.

23. gzip [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Gzip.

24. What is a CDN? [Електронний ресурс] – Режим доступу до ресурсу: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

25. About the PageSpeed Insights API [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/speed/docs/insights/v5/about.

26. Cloudflare [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Cloudflare.

27. .htaccess [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/.htaccess.

28. НПАОП 0.00-А.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці

29. НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці

30. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»

31. ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»

32. НПАОП 0.00.-1.28-10. Правила охорони праці під час експлуатації електронно-обчислювальних машин. Електробезпека.

33. ГОСТ 13109-97. Норми якості електричної енергії в системах електропостачання загального призначення.

34. ГОСТ 12.1.030-81. Захисне заземлення. Занулення.

35. ГОСТ 12.1.006-84. Електромагнітні поля радіочастот. Допустимі рівні на робочих місцях і вимоги до проведення контролю.

36. ДБН В.2.5-28:2015. Природне і штучне освітлення.

37. ДСаНПіН 3.3.2-007-98. Державні санітарні правила і норми.

38. НАПБ Б.03.002-2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою.

Додаток А

Слайди презентації

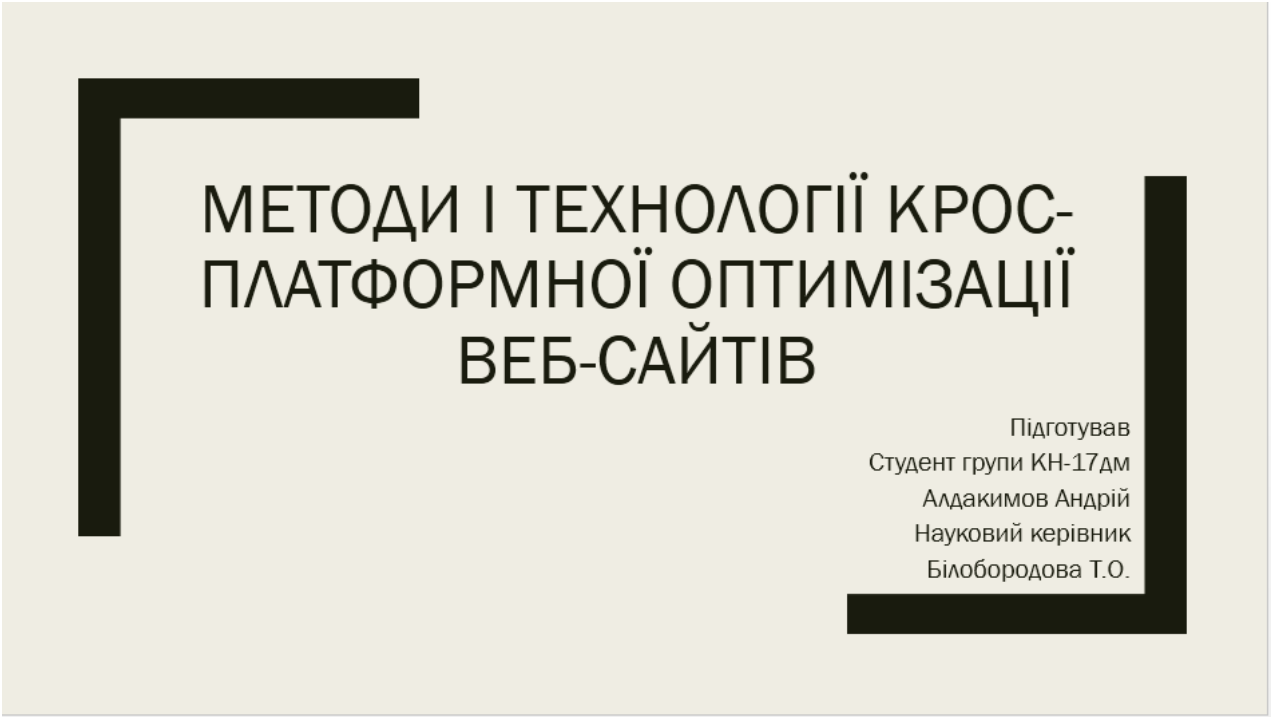


Рисунок А.1 – Титульний слайд

Рисунок А.2 – Мета і завдання дослідження

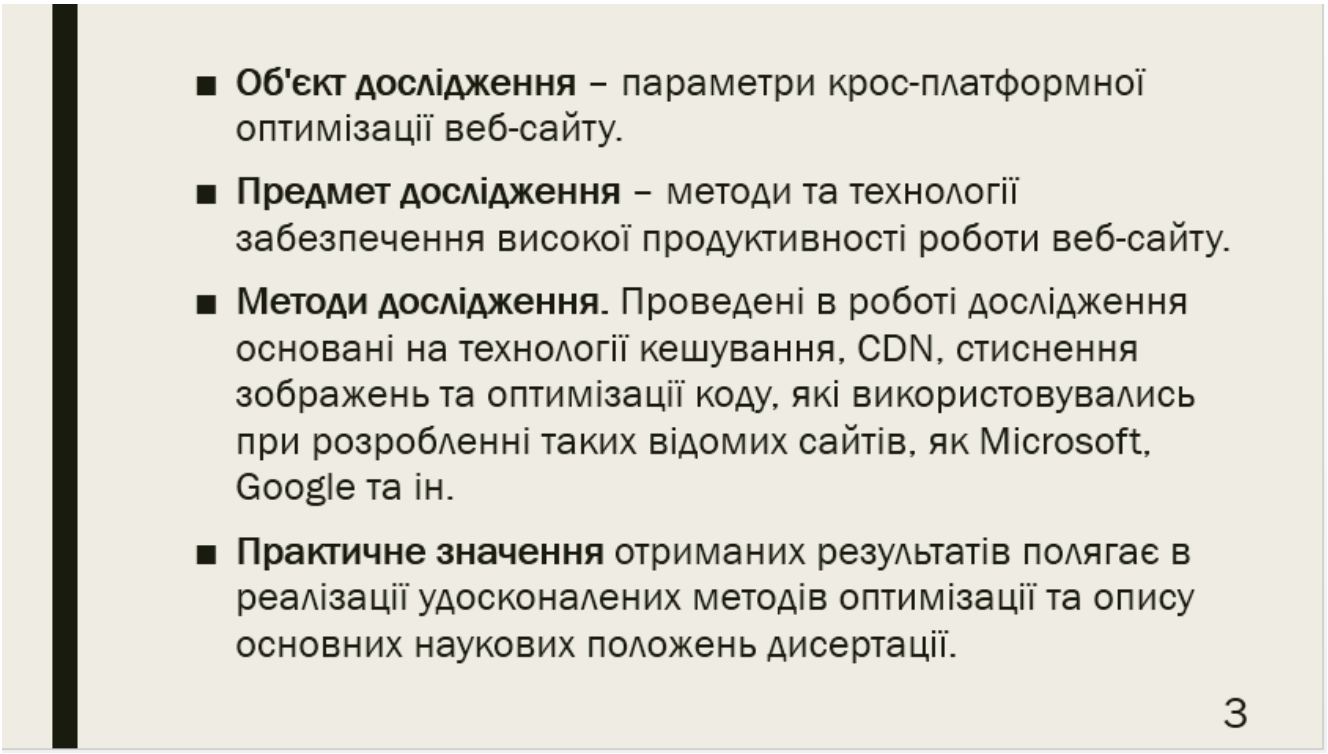


Рисунок А.3 – Об’єкт, предмет та методи дослідження

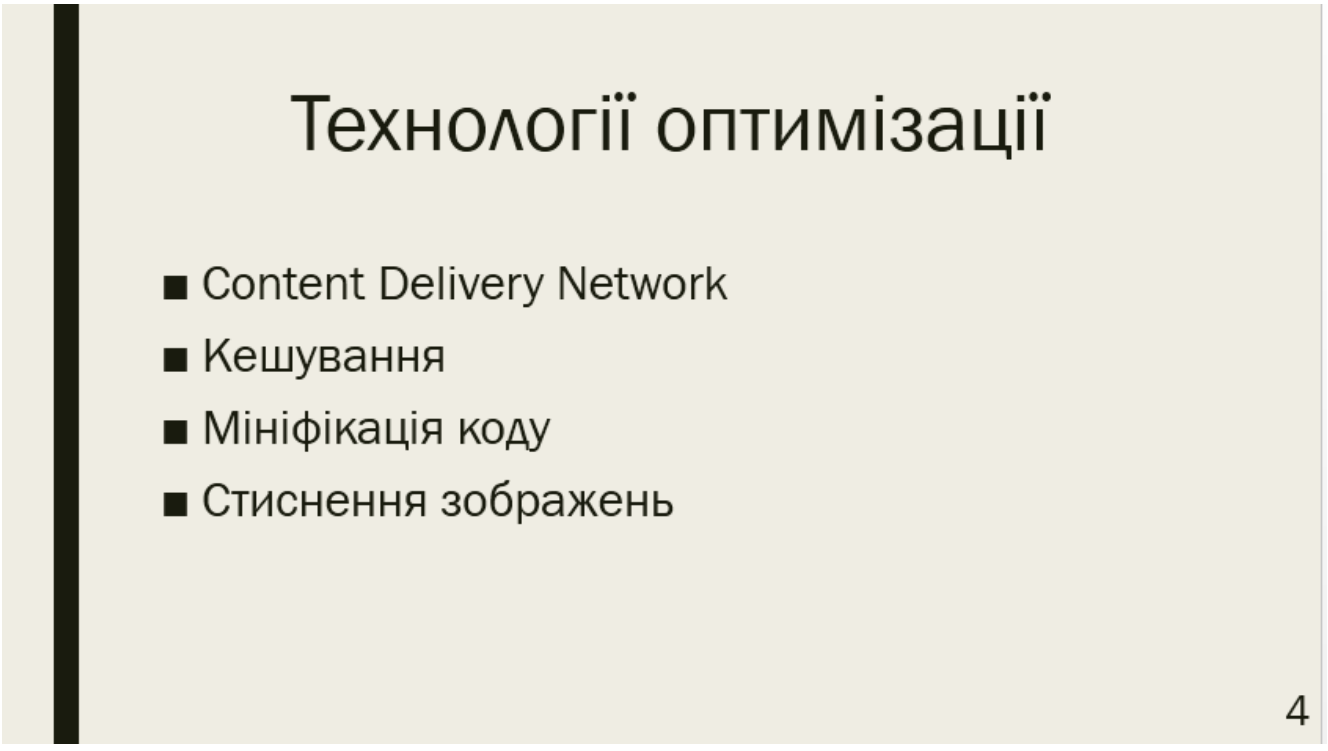


Рисунок А.4 – Існуючі методи та технології оптимізації

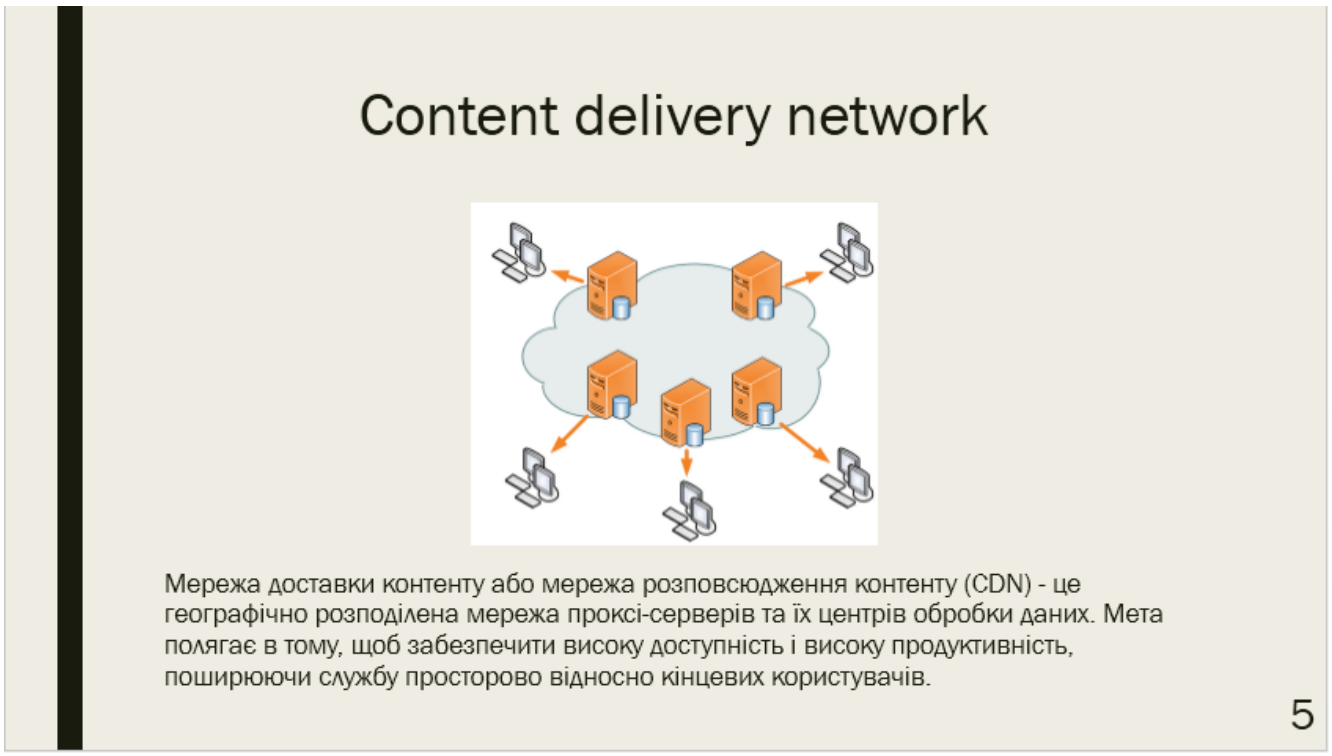


Рисунок А.5 – Принцип роботи CDN

Рисунок А.6 – Принцип роботи кешування

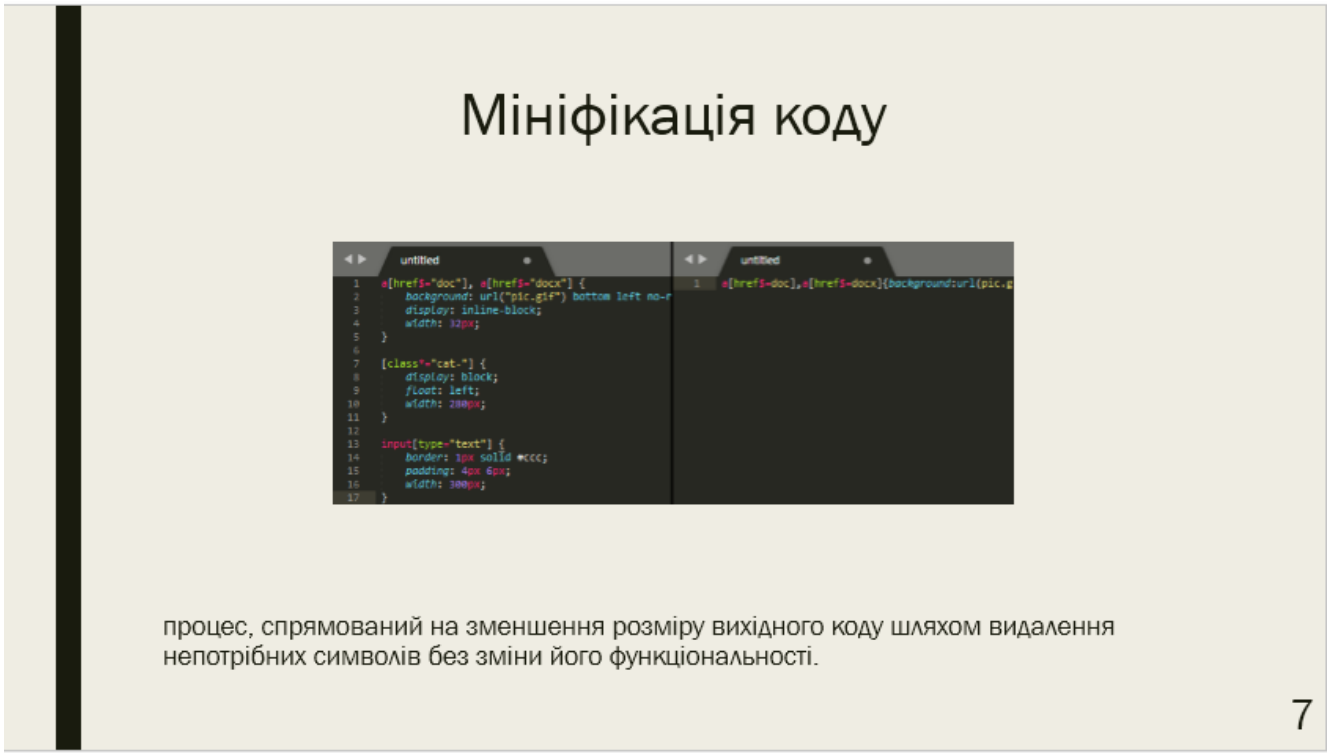


Рисунок А.7 – Результат мініфікації коду

Рисунок А.8 – Стиснення зображень

Рисунок А.9 – Завдання, що необхідно вирішити

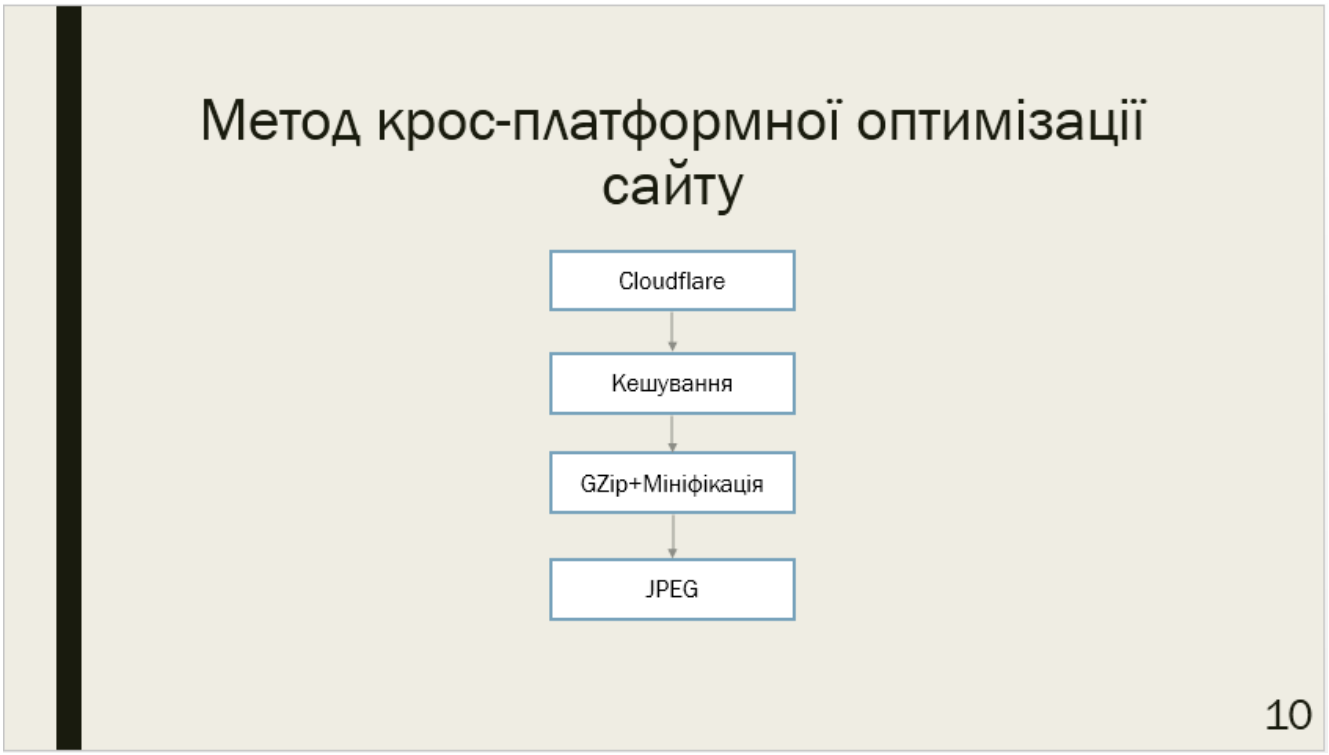


Рисунок А.10 – Схема запропонованого методу

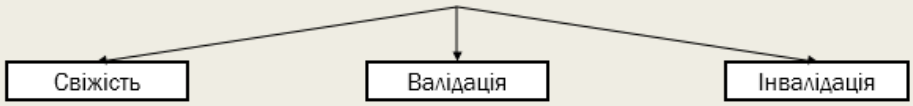
CDN

Cloudflare CDN за допомогою механізму зворотного проксінг nginx роздає сайти. Принцип роботи: у них розміщений дуже швидкий веб-сервер nginx який першим приймає на себе користувачів, і далі вже самостійно звертається до ваших ресурсів.

11

Рисунок А.11 – Cloudflare CDN

Технологія кешування



Свіжість дозволяє використовувати відповідь без повторної перевірки на вихідному сервері і може контролюватися як сервером, так і клієнтом.

Час старіння (freshnessLifetime) обчислюється на підставі декількох заголовків.

Термін дії обчислюється наступним чином:

expirationTime (термін дії) = responseTime + freshnessLifetime - currentAge (поточний вік),

де responseTime - це час отримання відповіді по годиннику браузера.

Валідація використана для перевірки того, чи кешована відповідь все ще гарна після того, як вона стане старою.

Інвалідація є побічним ефектом іншого запиту, який проходить через кеш.

Використано наступні елементи:

- Агресивно кешувальні елементи.
- Елементи з коротким терміном свіжості і необхідністю валідації.
- Елементи, що не кешуються взагалі.

12

Рисунок А.12 – Технологія кешування

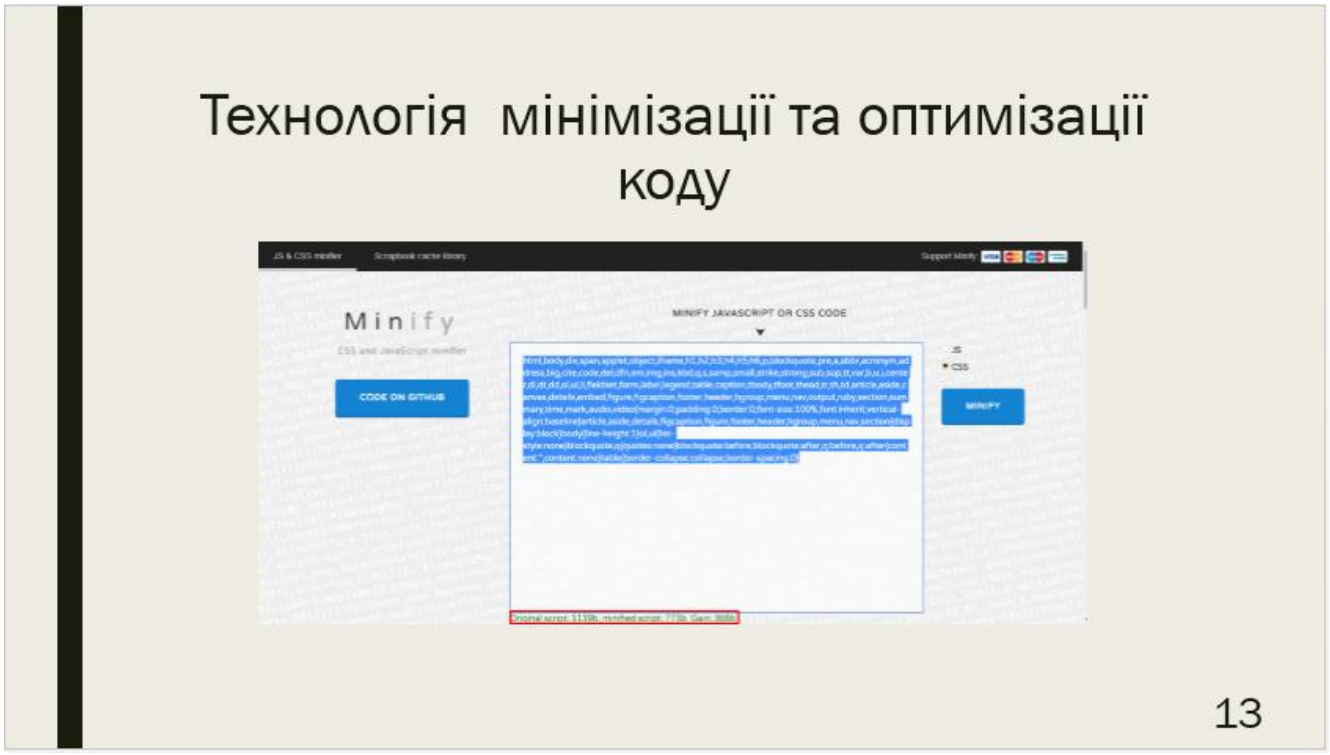


Рисунок А.13 – Мінімізація коду

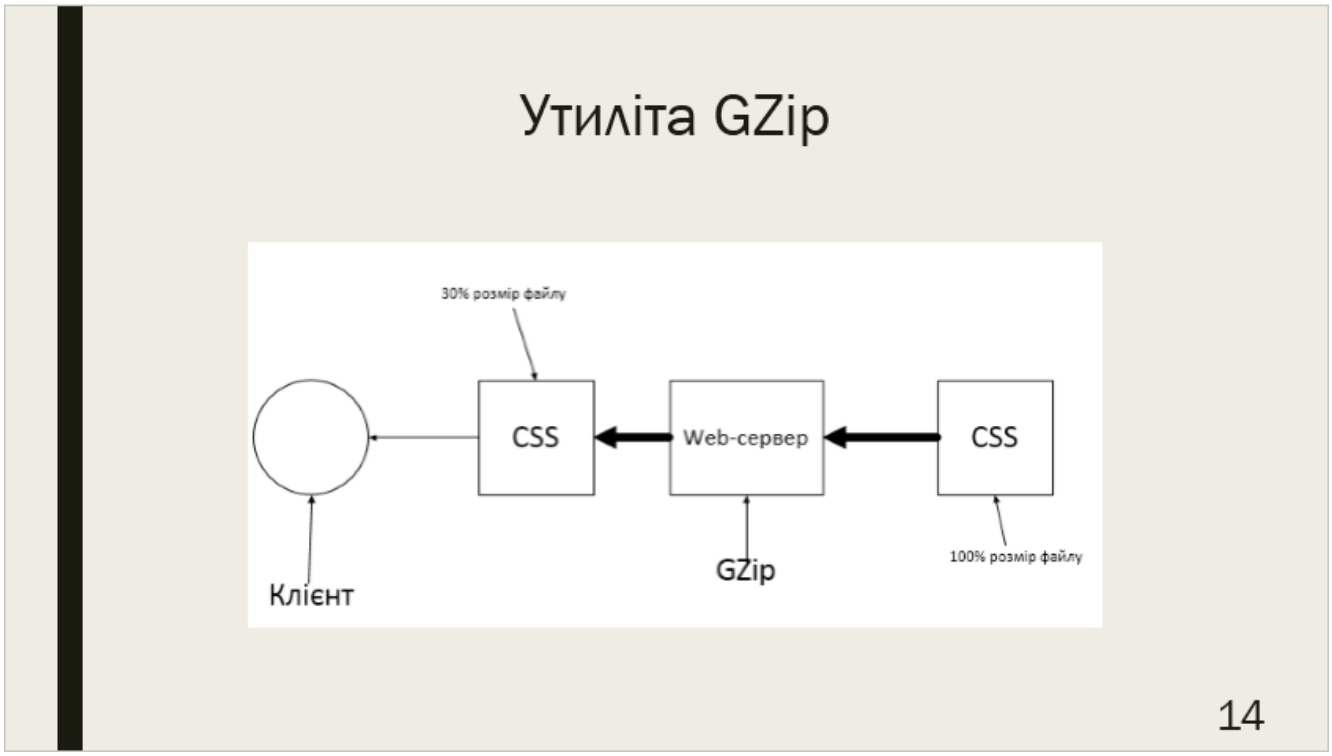


Рисунок А.14 – Утиліта GZip

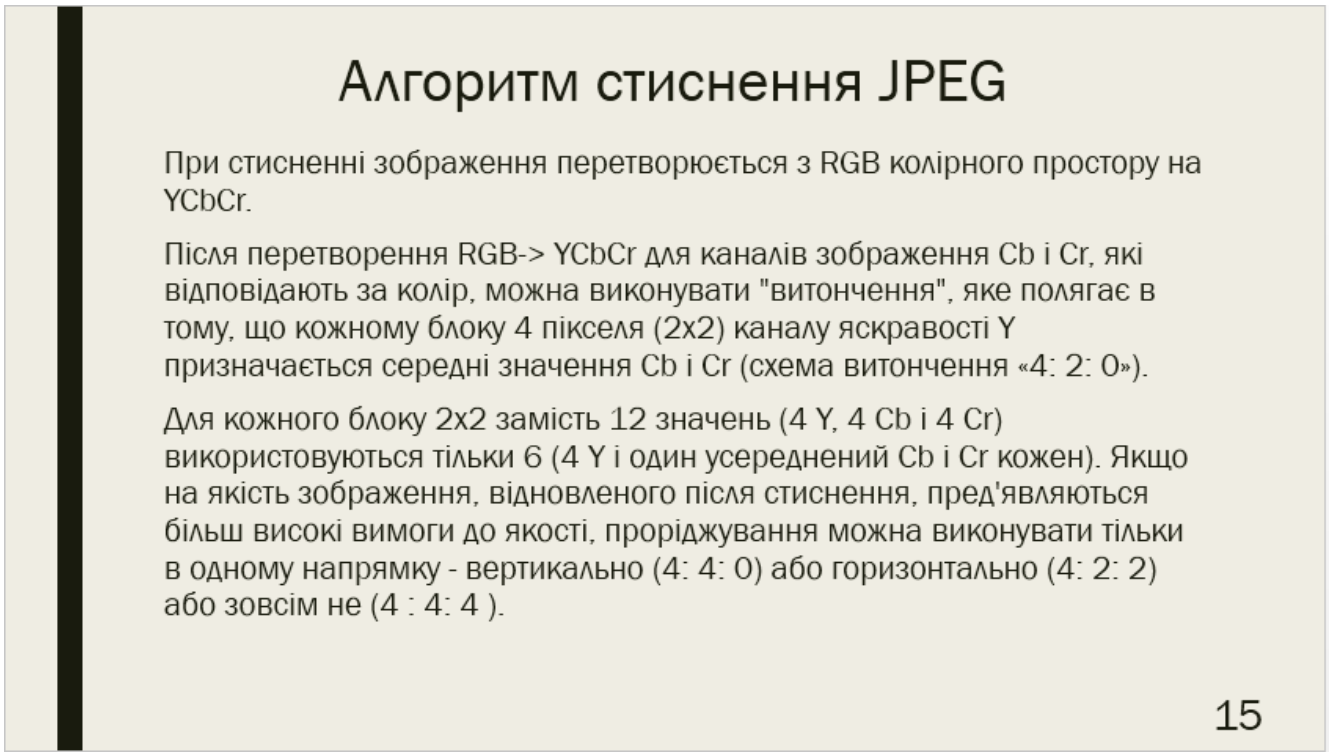


Рисунок А.15 –Алгоритм JPEG

Рисунок А.16 – Результат кешування

Рисунок А.17 – Результат мініфікації та GZip

Рисунок А.18 – Результат стиснення зображень

Результати оптимізації

Технології	До застосування методів		Запропонований метод	
	Перше завантаження	Наступні завантаження	Перше завантаження	Наступні завантаження
CDN	7,85	5,34	-	-
Кешування			7,85	4,98
Мініфікація			4,36	4,05
Стиснення зображень			5,04	4,69

19

Рисунок А.19 – Результати оптимізації

Висновки

- Досліджено відомі технології оптимізації сайту, визначені кращі з існуючих технологій для реалізації розроблюваного методу оптимізації.
- Визначена проблема оптимізації веб-сайту, яка полягає у відсутності універсального методу оптимізації.
- Встановлено, що основним напрямком роботи крос-платформної оптимізації є створення та реалізація універсального методу удосконаленої крос-платформної оптимізації сайту.
- Розроблено метод удосконаленої крос-платформної оптимізації сайту з використанням визначених технологій сучасної оптимізації.
- Отримані результати тестування методу, що демонструють високі показники збільшення швидкості завантаження майже на 65%.

20

Рисунок А.20 – Висновки

Додаток Б

Повний лістинг веб-сайту

Б.1 Головна сторінка

```
<!doctype html>
<html>
    <?php
    define('cdnURL',
'http://bonnie.ns.cloudflare.com.subdimensionsoft.eu/');
    ?>
<title>SubDimenisionSoft</title>
<head>
    <meta http-equiv='X-UA-Compatible' content='IE=Edge' />
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="keywords" content="SubDimenisionSoft, soft, game, games">
    <meta name="description" content="SubDimenisionSoft" />
    <meta name="author" content="admin" />
    <link   rel="stylesheet"   href="<?php   echo   cdnURL?>css/animate.css"
type="text/css">
    <link    rel="stylesheet"    href="<?php    echo    cdnURL?>css/reset.css"
type="text/css">
    <link    rel="stylesheet"    href="<?php    echo    cdnURL?>css/style.css"
type="text/css">
    <link    rel="stylesheet"    href="<?php    echo    cdnURL?>css/media.css"
type="text/css">
    <script async type="text/javascript">
        window.onload = function() {
            setTimeout(function() {
                $("center").animate({
                    opacity: "1",
                    opacity: "0"
                }, 500);

$("header").children().eq(0).removeClass("opacity").addClass("animated      delay2
fadeIn");

$("header").children().eq(1).removeClass("opacity").addClass("animated    delay2-2
fadeIn");
                $("#content-1  h1").removeClass("opacity").addClass("animated
delay1 fadeIn");
                $("#content-1   p").removeClass("opacity").addClass("animated
delay3 fadeInUp");
                $("#content-
2").children().removeClass("opacity").addClass("animated delay1 fadeIn");
                $("#content-
3").children().removeClass("opacity").addClass("animated delay1 fadeIn");
                $("#content-3-
mobile").children().removeClass("opacity").addClass("animated delay1 fadeIn");
                $("#content-
4").children().removeClass("opacity").addClass("animated delay1 fadeIn");

$("#write_us").children().removeClass("opacity").addClass("animated         delay1
fadeIn");
                $("footer
a").eq(0).removeClass("opacity").addClass("animated delay2-4 fadeIn");
                $("footer
a").eq(1).removeClass("opacity").addClass("animated delay2-6 fadeIn");
```

```
                    $("footer
a").eq(2).removeClass("opacity").addClass("animated delay2-8 fadeIn");
                    $("footer
a").eq(3).removeClass("opacity").addClass("animated delay3 fadeIn");
                    $("#videodiv").removeClass("opacity").addClass("animated
delay3 fadeIn");
                }, 100);
            };
        </script>
    </head>
    <body id="body">
        <div id="wrap">
            <header>
                <a href="#content-1" class="glitch-effect opacity">
                    <div class="glitch" data-text="SUB" id="logo">SUB</div>
                    <div          class="glitch"          data-text="DIMENSION"
id="logo">DIMENSION</div>
                    <div class="glitch" data-text="SOFT" id="logo">SOFT</div>
                </a>
                <a href="#write_us" rel=«nofollow» class="write_us opacity">
                    write us
                </a>
            </header>
            <center>
                <div                class="inTurnFadingTextG1">loading</div><div
id="inTurnFadingTextG_1"                    class="inTurnFadingTextG">.</div><div
id="inTurnFadingTextG_2"                    class="inTurnFadingTextG">.</div><div
id="inTurnFadingTextG_3" class="inTurnFadingTextG">.</div>
            </center>
            <div id="content" class="1">
                <div id="content-1">
                    <div class="main_text">
                        <h1             class="opacity">Software             and
game<br>development</h1>
                        <p   id="main"   class="opacity"   style="padding-left:
.3em;">scroll down to learn more</p>
                        <p id="mobil" class="opacity">touch at screen to learn
more</p>
                    </div>
                </div>
            </div>
            <div id="content" class="2">
                <div id="content-2">
                    <div class="main_text opacity">
                        <h1>about us</h1>
                        <ul class="menu" style="padding-left: .3em;">
                            <li>
                                <h2>Software development</h2>
                                <p>We    offer    services    in    integrated
implementation of software at every stage of product development from concept
creation to a ready-made product hardware and software components support</p>
                            </li>
                            <li>
                                <h2>Game development</h2>
                                <p>We develop multiplatform gaming projects,
projects in online and serious gaming segment, service products for internal and
commercial use. We are currently participating in men of war and call to arms
projects</p>
                            </li>
                        </ul>
                    </div>
                </div>
            </div>
            <div id="content" class="3">
```

```
                    <div id="content-3">
                        <div class="main_text opacity">
                            <h1>Our expertise</h1>
                            <ul class="menu" style="padding-left: .4em;">
                                <li>01<br>artificial intelligence</li>
                                <li>04<br>development of game missions</li>
                            </ul>
                            <ul class="menu">
                                <li>02<br>physical simulation</li>
                                <li>05<br>animation clips</li>
                            </ul>
                            <ul class="menu">
                                <li>03<br>3d-models and animation</li>
                            </ul>
                        </div>
                    </div>
                    <div id="content-3-mobile">
                        <!-- Для мобилок -->
                        <div class="main_text opacity">
                            <h1>our expertise</h1>
                            <ul class="menu">
                                <li>01 artificial intelligence</li>
                                <li>02 physical simulation</li>
                                <li>03 3d-models and animation</li>
                                <li>04 development of game missions</li>
                                <li>05 animation clips</li>
                            </ul>
                        </div>
                    </div>
                </div>
                <div id="content" class="4">
                    <div id="content-4">
                        <div class="main_text opacity">
                            <h1>contact us</h1>
                            <ul class="menu" style="padding-left: .3em;">
                                <li>
                                    <h2>location</h2>
                                    <p>ul. marsz. józefa piłsudskiego, 74, lok. 320,
wrocław, 50-020.</p>
                                </li>
                                <li>
                                    <h2>email</h2>
                                    <p>subdimensionsoft@gmail.com</p>
                                </li>
                            </ul>
                        </div>
                    </div>
                </div>
                <div id="content" class="5">
                    <div id="write_us">
                        <div class="main_text opacity">
                            <div class="discuss">
                                <h1>let's</h1>
                                <h1>discuss</h1>
                                <h1>the project</h1>
                            </div>
                            <script type="text/javascript">
                                var a = Math.ceil(Math.random() * 10);
                                var b = Math.ceil(Math.random() * 10);
                                var c = a + b
                                function DrawBotBoot() {
                                    m = document.write(a + " + " + b + "? ");
                                    document.getElementById("captcha").value = m;
                                    document.write("<input id='BotBootInput'/>");
```

```
                                }
                                function ValidBotBoot() {
                                    var                     d                     =
document.getElementById('BotBootInput').value;
                                    if (d == c) return true;
                                    return false;
                                }
                                function checkForm(form) {
                                    var  em  =  /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-
\.])+\.([A-Za-z]{2,4})$/;
                                    if
(!em.test(document.getElementById("email").value) || (ValidBotBoot() == false)) {
                                        if
(!em.test(document.getElementById("email").value)) {

document.getElementById("email").placeholder = "EMAIL FIELD IS REQUIRED";
                                            $("#email").addClass("email-1");
                                        } else {
                                            $("#email").removeClass("email-1");

document.getElementById("email").placeholder = "YOUR EMAIL";
                                        }
                                        if (ValidBotBoot() == false) {

$("#BotBootInput").addClass("BotBootInput");
                                        } else {

$("#BotBootInput").removeClass("BotBootInput");
                                        }
                                        return false;
                                    }
                                }
                            </script>
                            <div id="contact-wrapper">
                                <form          id="contactform"          method="post"
onsubmit="return checkForm(this)" action="send.php">
                                    <input   name="contactname"   class="contactname"
placeholder="YOUR NAME" />
                                    <input   name="email"   id="email"   class="email"
placeholder="YOUR EMAIL" />
                                    <textarea     name="message"     class="message"
placeholder="MESSAGE (OPTIONAL)"></textarea>
                                    <div class="captcha">
                                        <label id="captcha"></label>
                                        <script type="text/javascript">
                                            DrawBotBoot()
                                        </script>
                                    </div>
                                    <input  id="submit"  name="submit"  type="submit"
value="SEND" />
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
                <footer>
                    <ul class="menu">
                        <li>
                            <a class="opacity" rel=«nofollow» href="#content-2">
                                about
                            </a>
                        </li>
                        <li>
                            <a class="opacity" rel=«nofollow» href="#content-3">
```

```
                            our expertise
                        </a>
                    </li>
                    <li>
                        <a class="opacity" rel=«nofollow» href="#content-4">
                            contact
                        </a>
                    </li>
                </ul>
            </footer>
        </div>
        <div id="videodiv" class="overlay opacity">
            <video id="video-background" width="auto" height="auto" loop="true"
autoplay="autoplay" preload="auto" poster="<?php echo cdnURL?>img/bg1.jpg" muted>
                <source src="video/background.mp4" type = "video/mp4" />
                <source src="video/background.webm" type="video/webm" />
                <source src ="video/background.ogv" type= "video/ogg" />
            </video>
            <!--if   (!(navigator.userAgent.match(/(iPod|iPhone|iPad|Android)/)))
{-->
                        <!--document.write('<video          id="video-background"
width="auto"   height="auto"   loop="true"   autoplay="autoplay"   preload="auto"
poster="img/bg1.jpg"><source        src="video/background.mp4"        type        =
"video/mp4"></source><source                              src="video/background.webm"
type="video/webm"></source><source      src      ="video/background.ogv"      type=
"video/ogg"></source></video>');-->
                <!--}    else    document.write('<video    id="video-background"
width="auto"   height="auto"   loop="true"   autoplay="autoplay"   preload="auto"
poster="img/bg.jpg"></video>');-->
        </div>
        <!-- JavaScript -->
        <script src="https://code.jquery.com/jquery-latest.js"></script>
        <script src="<?php echo cdnURL?>js/jquery.easing.1.3.js"></script>
        <script src="<?php echo cdnURL?>js/jquery.mousewheel.js"></script>
        <script          type="text/javascript"          src="<?php          echo
cdnURL?>js/jquery.touchSwipe.min.js"></script>
        <script src="<?php echo cdnURL?>js/myjquery.js"></script>
        <script>
            function windowSize() {

                if   (navigator.userAgent.match(/(iPod|iPhone|iPad|Android)/)   ||
$(window).width() <= '960') {
                    $("footer .menu, #content-3, #main").hide();
                    $("#content-3-mobile, #mobil").show();
                } else {
                    $("footer .menu, #content-3, #main").show();
                    $("#content-3-mobile, #mobil").hide();
                }
            }
            $(window).on('load resize', windowSize);
        </script>
    </body>
    </html>
```

Б.2 Файл стилей

```
html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,a
,abbr,acronym,address,big,cite,code,del,dfn,em,img,ins,kbd,q,s,samp,small,strike,s
trong,sub,sup,tt,var,b,u,i,center,dl,dt,dd,ol,ul,li,fieldset,form,label,legend,tab
le,caption,tbody,tfoot,thead,tr,th,td,article,aside,canvas,details,embed,figure,fi
```

```
gcaption,footer,header,hgroup,menu,nav,output,ruby,section,summary,time,mark,audio
,video{margin:0;padding:0;border:0;font-size:100%;font:inherit;vertical-
align:baseline}article,aside,details,figcaption,figure,footer,header,hgroup,menu,n
av,section{display:block}body{line-height:1}ol,ul{list-
style:none}blockquote,q{quotes:none}blockquote:before,blockquote:after,q:before,q:
after{content:'';content:none}table{border-collapse:collapse;border-
spacing:0}@font-face{font-family:'barlow_condensedmedium';font-display:
auto;src:url(../fonts/barlowcondensed-medium-
webfont.eot);src:url('../fonts/barlowcondensed-medium-webfont.eot?#iefix')
format('embedded-opentype'),url(../fonts/barlowcondensed-medium-webfont.woff2)
format('woff2'),url(../fonts/barlowcondensed-medium-webfont.woff)
format('woff'),url(../fonts/barlowcondensed-medium-webfont.ttf)
format('truetype'),url('../fonts/barlowcondensed-medium-
webfont.svg#barlow_condensedmedium')            format('svg');font-weight:500;font-
style:normal}*{margin:0;padding:0;border:none}body{width:500vw;font-
family:"barlow_condensedmedium";overflow:hidden;font-weight:500;text-
transform:uppercase;color:rgb(255,255,255);background-color:#000b16;-moz-
background-size:cover;-webkit-background-size:cover;-o-background-
size:cover;background-size:cover}h1{font-size:7.8rem;width:auto;padding:0  0  .6em
0}h2{font-size:1.41rem;color:rgb(255,255,255);-webkit-margin-before:0;-webkit-
margin-after:0;-webkit-margin-start:0;-webkit-margin-end:0;padding-
bottom:.5em}li{font-size:1.41rem;padding:0  3em  2em  0;float:left;width:auto;line-
height:1.34em}p{font-size:1.41rem;color:rgba(255,255,255,.22);-webkit-margin-
before:0;-webkit-margin-after:0;line-height:1.34em}a{text-
decoration:none;outline:none;color:rgb(255,255,255);transition:.5s;font-
size:1.41rem}.opacity{opacity:0}ul.menu{list-style:none;float:left}ul.menu      li
a{text-decoration:none;outline:none}.main_text{margin:12em          0          0
18em}header{width:100%;top:3.125rem;position:fixed;z-index:1000;font-
size:1.3em}header    a.glitch-effect{float:left;margin:0    0    0    3rem}header
a.write_us{float:right;margin:0                      3rem                      0
0}.glitch{color:white;position:relative}#content{float:left;width:20%;height:100%}
#content-2     ul.menu     li{padding:0     5em     0     0;width:17em}#content-3
ul.menu{width:23em}#content-4
ul.menu{width:30em}footer{bottom:3.125rem;position:fixed;z-
index:1000;width:100%;font-size:1.3em}footer   ul.menu   li{padding:0   .7em   0
0;margin:0                                  0                                  0
3rem}.active{color:gray;transition:.5s}.overlay{width:100%;height:100%;display:blo
ck}#videodiv{position:fixed;overflow:hidden;z-index:-1}#video-
background{position:fixed;width:100%;height:100%}@supports                (object-
fit:cover){video{width:100%;height:100%;object-fit:cover}}@keyframes        noise-
anim{0%{clip:rect(100px,9999px,18px,0)}5%{clip:rect(59px,9999px,45px,0)}10%{clip:r
ect(55px,9999px,94px,0)}15%{clip:rect(80px,9999px,51px,0)}20%{clip:rect(33px,9999p
x,72px,0)}25%{clip:rect(9px,9999px,88px,0)}30%{clip:rect(12px,9999px,11px,0)}35%{c
```

```
lip:rect(81px,9999px,24px,0)}40%{clip:rect(8px,9999px,89px,0)}45%{clip:rect(50px,9
999px,27px,0)}50%{clip:rect(81px,9999px,22px,0)}55%{clip:rect(96px,9999px,78px,0)}
60%{clip:rect(41px,9999px,50px,0)}65%{clip:rect(6px,9999px,68px,0)}70%{clip:rect(3
3px,9999px,92px,0)}75%{clip:rect(36px,9999px,28px,0)}80%{clip:rect(43px,9999px,51p
x,0)}85%{clip:rect(20px,9999px,47px,0)}90%{clip:rect(93px,9999px,23px,0)}95%{clip:
rect(70px,9999px,82px,0)}100%{clip:rect(69px,9999px,33px,0)}}.glitch#logo:after,.g
litch:hover:after{content:attr(data-text);position:absolute;left:2px;text-shadow:-
2px                                                                          0
#07f2f6;top:0;color:white;overflow:hidden;clip:rect(0,900px,0,0);animation:noise-
anim     3s     infinite     linear     alternate-reverse}@keyframes     noise-anim-
2{0%{clip:rect(54px,9999px,24px,0)}5%{clip:rect(88px,9999px,38px,0)}10%{clip:rect(
86px,9999px,26px,0)}15%{clip:rect(19px,9999px,50px,0)}20%{clip:rect(73px,9999px,26
px,0)}25%{clip:rect(71px,9999px,72px,0)}30%{clip:rect(68px,9999px,96px,0)}35%{clip
:rect(9px,9999px,38px,0)}40%{clip:rect(2px,9999px,75px,0)}45%{clip:rect(59px,9999p
x,86px,0)}50%{clip:rect(35px,9999px,43px,0)}55%{clip:rect(54px,9999px,98px,0)}60%{
clip:rect(52px,9999px,40px,0)}65%{clip:rect(23px,9999px,56px,0)}70%{clip:rect(99px
,9999px,99px,0)}75%{clip:rect(94px,9999px,87px,0)}80%{clip:rect(23px,9999px,95px,0
)}85%{clip:rect(74px,9999px,24px,0)}90%{clip:rect(71px,9999px,5px,0)}95%{clip:rect
(14px,9999px,1px,0)}100%{clip:rect(22px,9999px,78px,0)}}.glitch#logo:before,.glitc
h:hover:before{content:attr(data-text);position:absolute;left:-2px;text-shadow:2px
0    #f0f;top:0;color:white;overflow:hidden;clip:rect(0,900px,0,0);animation:noise-
anim-2            4s            infinite            linear            alternate-
reverse}.discuss{width:37em;float:left}.discuss  h1,.form_done  h1{padding:0  0  0
0}#contact-wrapper{width:27em;float:left;margin-top:1.3em;margin-
left:6em}.contactname,.email{height:3em;background:url(../img/nameemail_zip.webp)}
.message{padding-
top:1em;height:10em;background:url(../img/message_zip.webp)}.contactname,.email,.m
essage{width:27em;font-family:inherit;color:#fff;margin-bottom:1em;text-
indent:1.5em;font-size:1em}.email-1{border:.1em          solid          red;text-
indent:3.5em;background:url(../img/nameemail_zip.webp)
repeat,url(../img/alert.png)    1.2em    1em    no-repeat}#submit{float:left;font-
family:inherit;font-
size:1.2em;color:rgb(25,16,38);width:8.3em;height:2.8em}.captcha{float:right;font-
family:inherit;color:#fff;font-size:2.7em}#BotBootInput{font-
family:inherit;color:#fff;font-size:1em;width:1.5em;text-align:center;margin-
left:.3em;background:url(../img/nameemail_zip.webp)}.BotBootInput{border:.03em
solid                          red}::-webkit-input-placeholder{color:#efefef}::-moz-
placeholder{color:#fff}:-moz-placeholder{color:#fff}:-ms-input-
placeholder{color:#fff}::placeholder{color:#fff}.email-
1::placeholder{color:red}.return_home{font-family:inherit;font-
size:1.2em;width:10em;height:2.5em}.form_done{width:100vw;height:100%;color:rgb(25
5,255,255);text-align:center;margin-top:15em;position:fixed}.form_done
h2{padding:1.5em    0}center{width:100%;height:100%;font-size:1.5em;display:inline-
```

```
flex;flex-direction:row;justify-content:center;align-
items:center;position:fixed}.inTurnFadingTextG1{color:rgb(255,255,255)}.inTurnFadi
ngTextG{color:rgb(255,255,255);animation-name:bounce_inTurnFadingTextG;-o-
animation-name:bounce_inTurnFadingTextG;-ms-animation-
name:bounce_inTurnFadingTextG;-webkit-animation-name:bounce_inTurnFadingTextG;-
moz-animation-name:bounce_inTurnFadingTextG;animation-duration:2.09s;-o-animation-
duration:2.09s;-ms-animation-duration:2.09s;-webkit-animation-duration:2.09s;-moz-
animation-duration:2.09s;animation-iteration-count:infinite;-o-animation-
iteration-count:infinite;-ms-animation-iteration-count:infinite;-webkit-animation-
iteration-count:infinite;-moz-animation-iteration-count:infinite;animation-
direction:normal;-o-animation-direction:normal;-ms-animation-direction:normal;-
webkit-animation-direction:normal;-moz-animation-
direction:normal}#inTurnFadingTextG_1{animation-delay:0.75s;-o-animation-
delay:0.75s;-ms-animation-delay:0.75s;-webkit-animation-delay:0.75s;-moz-
animation-delay:0.75s}#inTurnFadingTextG_2{animation-delay:0.9s;-o-animation-
delay:0.9s;-ms-animation-delay:0.9s;-webkit-animation-delay:0.9s;-moz-animation-
delay:0.9s}#inTurnFadingTextG_3{animation-delay:1.05s;-o-animation-delay:1.05s;-
ms-animation-delay:1.05s;-webkit-animation-delay:1.05s;-moz-animation-
delay:1.05s}@keyframes
bounce_inTurnFadingTextG{0%{color:rgb(0,0,0)}100%{color:rgb(255,255,255)}}@-o-
keyframes
bounce_inTurnFadingTextG{0%{color:rgb(0,0,0)}100%{color:rgb(255,255,255)}}@-ms-
keyframes
bounce_inTurnFadingTextG{0%{color:rgb(0,0,0)}100%{color:rgb(255,255,255)}}@-
webkit-keyframes
bounce_inTurnFadingTextG{0%{color:rgb(0,0,0)}100%{color:rgb(255,255,255)}}@-moz-
keyframes
bounce_inTurnFadingTextG{0%{color:rgb(0,0,0)}100%{color:rgb(255,255,255)}}
```