Міністерство освіти і науки України
Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра комп'ютерних систем

Рівень вищої освіти_____другий (магістерський)_____
Спеціальність__123 Комп'ютерна інженерія_____
Спеціалізація/ освітня програма ___Спеціалізовані ком'ютерні системи_____

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
“____” _____20___року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____Шеін Микола Анатолійович_____
(прізвище, ім’я, по батькові)

1. Тема роботи_____Дослідження методів підвищення продуктивності веб-додатків
_____
_____
Керівник роботи_____Степень Павел Вячеславович, доцент_____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ректора ОНПУ від “___”_____20__року №___
2. Зміст роботи
Розділ 1. Аналітичний огляд_____
Розділ 2. Дослідження методів для збільшення продуктивності веб-додатків
_____
Розділ 3. Дослідження швидкості завантаження_____
_____
_____
3. Перелік ілюстративного матеріалу
_____
_____
_____
_____
_____
_____
_____
_____

4. Консультанти розділів роботи

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

5. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка

Здобувач вищої освіти _____ Шеін М.А.

Керівник роботи _____ Ступень П.В.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HTML– HyperText Markup Language

XML – eXtensible Markup Language

CSS – Cascading Style Sheet

БД – База даних

БП – Бюджет продуктивності

CDN – Content Delivery Network

PHP - Personal Home Page

WWW – World Wide Web

OSI - Open Systems Interconnection

LL – Lazy Load

SPA – Single Page Aplication

ВСТУП

На сьогоднішній день, інформаційні сервіси є невідмінною рисою нашого часу. Інформаційні сервіси можна розглядати як: джерела інформації, спілкування, але і так же як інструмент організації та просування електронного бізнесу. Найбільш вираженим прикладом цих сервісів є WEB-додатки.

З їх розвитком, значно збільшилось споживання інтернет трафіку. Використання різних сайтів соціальних мереж, розміщення файлів на обачному сховищу даних, перегляд потокового відео стало в наш час дуже популярним. Це призводить до того, що провайдер повинен представляти, гаразду більшу пропускну спосібність інтернету, щоб закрити потрібності кінцевих користувачів. Але все одно, багато людей страдає від поганого доступу до мережі, та на низьку загрузку веб-додатків, та різних сервісів.

Забезпечення необхідної продуктивності веб-додатків, дуже важлива річь. Даний параметр може впливати одразу на три речі, які можна виділити:

- пошукова оптимізація. Це як пошукова система рекомендує ваш сайт, и показує його, при тангентних запитах користувача;
- опит користувача, або UX. Це опит та емоції, які викликає конкретний продукт. Емоції від продукту можуть бути різними. Можливо користувач відчує радість, та легкість при використані продукту, або негатив тому що користувачу не подобається як виглядає, або працює сайт. Основні пошукові системи, використовують різні методи, щоб визначити зручність в використанні певного веб-додатку;
- економія використанні ресурсів серверу, та користувача. Щоб запобігти зростання навантаження на сервери, можна використати наступні методи балансування навантаження які відповідають рівням моделі OSI: мережевому, транспортному та прикладному рівнів. Існують також інші методи збільшення швидкодії роботи веб-додатків;

Ці всі речи переслідують одну головну ціль. Тримати певний рівень конверсії на сайті. Це процент трафіку, який виконує якусь дію. У сайту може бути дуже гарний дизайн, та широкий функціонал можливостей які не будуть мати значення якщо він буде довго завантажуватись. Сучасні користувачі, все менш терпеливі, кому кожна секунда має вагу. Рівень їх нетерплячості буде рости разом з збільшенням долі мобільних пристроїв. Якщо це інтернет магазин, користувач переглядає товар або робить замовлення, і це повинному бути максимально легко, швидко, або коли користувач підписується на ваш сайт з новинами, щоб частіше переглядати новини. Якщо веб-додаток повільний, користувачі не зможуть завершити ці дії, що призведе до покидання даного додатку та втрати потенціального прибутку.

Для вирішення цих питань, у даній магістерські роботі будуть розглянуті апаратні рішення, для забезпечення рівня продуктивності веб-додатків. **Актуальною задачею** э оптимізація швидкості завантаження, звичайних та “важких сайтів”, за допомогою нового методу, праці з зображеннями.

Метою дослідження – є розробка нового методу праці з зображеннями, що дозволить збільшити швидкість початкового завантаження сторінки, за допомогою зменшення ваги зображень, та зробити веб-додаток більш інтерактивним. Це дозволить влучити даний метод, во всі готові рішення, шляхом змінення коду сторінки.

Для досягнення мети, потрібно вирішити наступні завдання дослідження:

- зробити аналіз існуючих методів роботи з картинками, які при результаті збільшать продуктивність, і швидкість завантаження сторінки;
- обґрунтування вибір даного методу, його переваги і недоліки і зрівнянні з іншими;
- здійснення реалізації даного методу, проведення тестування і аналіз отриманих результатів, зробити оцінку ефективності даного методу.

Об’єкт дослідження – методи роботи з зображеннями.

Предмет дослідження – процес оптимізації зображень.

Наукова новизна – розроблено методу праці з зображеннями, який на рівні тегу використовує векторний формат зображення, SVG.

Практичне значення роботи полягає у реалізації даного методу на реальному сайті, подальша оцінка його продуктивності та швидкодії завантаження.

В перспективі метод створений у роботі, може бути покращена шляхом залучення нових більш ефективних мало-важких форматів зображення.

За результатами дослідження, була підготовлена публікація в фаховий журнал.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД

1.1. Мета магістерської роботи

Метою даної магістерської роботи є, дослідження методів збільшення продуктивності праці веб-додатків. Дана тема є актуальною тому, що кожна людина користується інтернетом, і заходить на різні веб-додатки. Збільшення продуктивності, цікава для двох сторін, власників веб-додатків, і для користувачів.

Тривалий час завантаження веб-додатку негативно впливає на тривалість сеансу клієнта. Якщо час завантаження вашого веб-додатку надто тривалий, користувач не стане чекати його завантаження та покине сторінку. Ця дія дасть негативну оцінку для вашого сайту. Після цього пошукові системи будуть зменшувати рейтинг видачи веб-додатку. Із зменшенням рейтингу, сайт може втратити потенціальних користувачів, зменшитись кількість продаж, якщо це інтернет магазин, тощо.

Методи збільшення продуктивності, можуть буди трьох напрямків:

- програмні методи;
- апаратні методи;
- мережеві методи.

До програмних методів можна віднести, дослідження основних ресурсів сайту. Основні методи роботи с JavaScript або іншими мовами програмування, мовою розмітки HTML, таблицями стилів CSS. Роботу с картинками, так як вони є невід’ємною частиною кожного веб-додатку. Також до програмних методів, віднесемо дуже важливу річ, як кешування.

До апаратних методів, можна віднести швидкодії роботи серверу, на якому розміщується веб-додаток, та швидкодію роботи пристрою клієнта. За рахунок швидкості, відбувається рендерінг веб-додатка на стороні клієнту.

Мережеві методи, включають у себе багато різних моментів. Із основних можна виділити використання протоколу HTTP/ 2, дослідження зміни часу завантаження при зменшенні кількості запитів DNS. Також можна віднести швидкість роботи мережі, у кінцевого користувача, та серверу на якому лежить веб-додаток.

Саме тому, в даній магістерській роботі будуть досліджені методи збільшення продуктивності , цих трьох напрямків, щоб розв’язати наступні проблеми:

- своєчасну оптимізацію веб сайту;
- зменшення часу завантаження сайту;
- зменшення виробітки ресурсу, приладу клієнта, та її оптимізація .

1.2. Дослідження методів роботи з зображеннями

Час доступу до веб-додатку залежить від загального розміру контенту, який буде загружатись із хост-серверу. Високоякісні зображення вносять найбільший внесок у розмір сторінки сайту, знижаючи швидкість завантаження сторінки. Частіше всього трапляється так, що контент завантажився, а зображення все ще не загрузились. За цей час користувачі можуть не дочекатись кінця загрузки якоїсь сторінки, та покинути її.

Щоб зменшити негативний вплив різних зображень вашого веб-додатку, на швидкість роботи сайту, використовують наступні методи оптимізації:

Вибір формату зображення. Детальну статистику можна розглянути в таблиці 1.1 Формати зображені, від цілі користування.

1.2.1. Стискання зображень

Стискання зображення повинно бути розумним. Шукати компроміс між якістю зображення, та його розміром. Стискання на 60-70% э гарним

показником для форматів JPEG, WebP. Для цього можна використати наступні інструменти TinyPNG, ImageOptimizer або JPEGmini.[1]

1.2.2. Зображення низької якості

Замість картинки високої якості, можна використовувати неякісне, розмите зображення низької якості . Це дасть користувачу уявлення, що він може чекати від даного зображення , та що воно ще завантажуйтеся.

1.2.3. Метод ледачого завантаження зображення

Lazy Load – це метод, коли зображення за кордоном виділеної області сторінки не завантажується, поки зображення не приблизиться к краю видимої області. Перевага даного методу в тому, що за час першої загрузки веб-додатку, буде загружено лише мала частина зображень. Всі інші зображення, будуть загружатись по необхідності.

Приклад, як воно може виглядати. На нашому сайті, який завантажуються є чотири картинки, загальною вагою 6 MB. Розглянемо перший випадок.

У цьому випадку, всі зображення, будуть завантажуватись як звичайно. І клієнту потрібно бути завантажити за один раз 6мб тільки зображень. У сучасних реаліях це багато, і такі речі потрібно оптимізувати.

```
<img align="center" src="cat.jpg" alt=".."/>
<img align="center" src="cat-1.jpg" alt=".."/>
<img align="center" src="cat-2.jpg" alt=".."/>
<img align="center" src="cat-3.jpg" alt=".."/>
```

Приклад завантаження зображень, без Lazy load.

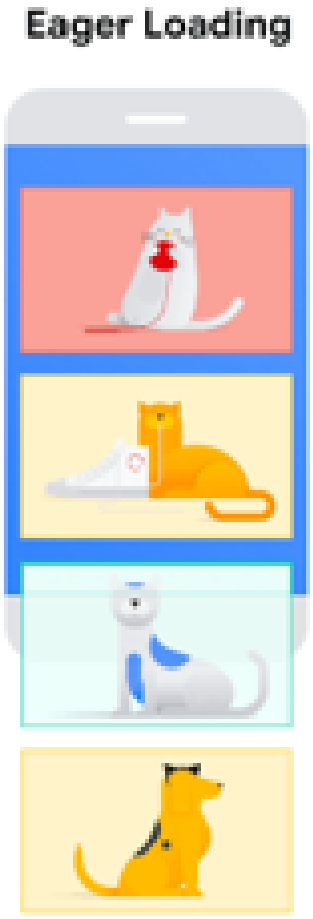

Рисунок 1.1 – Приклад завантаження зображень, на сайті

Праця, с іншими картинками, прийнявши метод завантаження ледачого завантаження, або Lazy Load.

```
<img data-src="cat-1.jpg" loading="lazy" alt=".." class="lazyload"/>
<img data-src="cat-2.jpg" loading="lazy" alt=".." class="lazyload"/>
<img data-src="cat-3.jpg" loading="lazy" alt=".." class="lazyload"/>
<img data-src="cat-4.jpg" loading="lazy" alt=".." class="lazyload"/>
```

При використанні data-src, зображення не буде одразу завантажене. Атрибут loading використовується для того, щоб вказати бібліотеці на зображення, з яким потрібно працювати. Приклад завантаження можна побачити на рисунку 1.2.

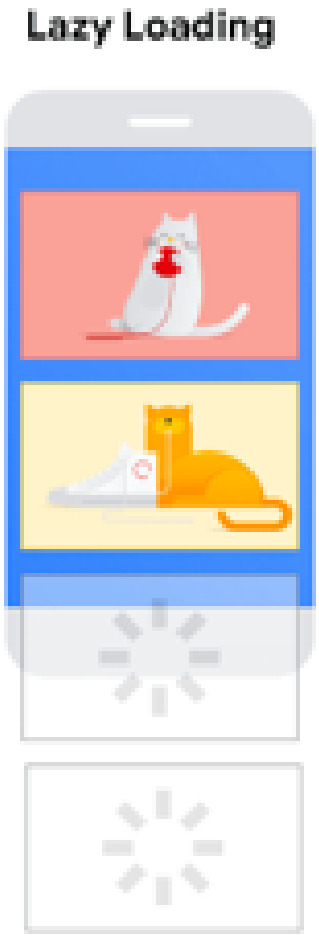

Рисунок 1.2 – Приклад завантаження картинок за допомогою Lazy Load

При використанні даного методу, клієнт завантажте спочатку ті картинки, які є в зоні видимості, у даному випадку це тільки 2/4 зображень. Завдяки цьому загальна вага, яку загрузить користувач, буде 3.04мб. Це в два рази менше, ніж завантажувати весь об’єм одразу. При використані атрибуту loading, браузер юзера запросить перші 2кб зображення у сервера. В цих 2кб даних, лежить інформація о розмірах картинки. Це дозволяє браузеру генерувати місця заповнювачі, для зображень , які ще не попали у зону видимості.

Існує багато бібліотек JavaScript, які доступні для добавлення Lazy Load на ваш веб-додаток, або веб-сайт. Наприклад LazySites, React Lazy Load.[2]

1.3. Ділення JavaScript коду

Можна виділити одну дуже корисну оптимізацію, для веб додатку, це ділення JavaScript коду. Звернувши увагу на дану оптимізацію, виділимо переваги, які можна отримати за її допомоги:

1 Інтерактивність. Користувач повинен знати, що при завантаженні сайту, на ньому щось відбувається, при різних завантаженнях.

2 Мінімізація. Завантаження мінімального коду, щоб якомога збільшити продуктивність.

3 Виконання коду, рендерінг. Максимально швидко виконати рендерінг коду, на стороні клієнта.

Основна ідея даного методу, це розділення великих зв’язок коду, на більш дрібні. Звернувши увагу на JavaScript, можна зрозуміти що він має дві вартості. Перша – це завантаження JavaScript, а друга, це його виконання.

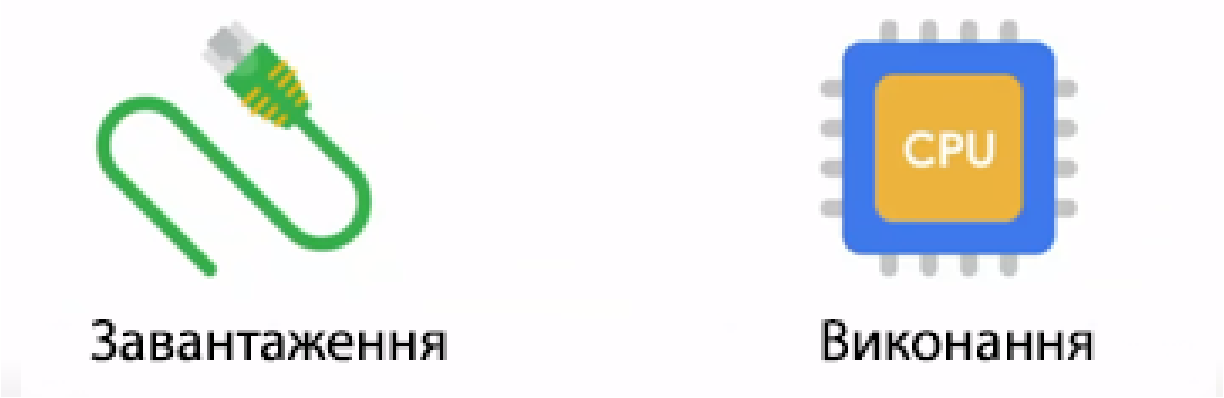

Рисунок 1.3 – Демонстрація вартості JavaScript.

На сьогоднішній день, завантаження JavaScript коду є головною перешкодою, для досягнення оптимальної продуктивності роботи веб додатку. Час завантаження, дуже критичний для повільних мереж, таких як 2G, та 3G. Тому важливо, зоб кількість завантажених файлів по мережі була якомога менша.

Що відноситься до виконання JavaScript коду, є пряма залежність від процесора приладу, з якого відбувається виконання коду. Виконання коду може бути критичних, до старих або дуже бюджетних приладах. Проведемо тестування.

При розгляді JavaScript, можно сказати, що він впливає на дуже головну метрику, таку як час до появи інтерактивності. Дуже багато додатків, у себе мають більше 600 КБ скриптів. Така кількість скриптів буде відштовхувати значний процент користувачів, тому що вони не будуть чекати завантаження інтерфейсу. А ті користувачі, які чекають на завантаження, будуть відчувати значні лаги на веб-додатку.

Тому, одразу варто задати бюджет продуктивності, для обмеження розміру JS файлу, який составе 150 КБ. Після завантаження сторінки, вона повинна реагувати на дії користувача, та бути інтерактивною.

За обробку дій з користувачем, генерацію DOM, відповідає браузер. Розглянемо популярну архітектуру SPA [3].

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="/styles.css">
<script src="/app.js" async></script>
</head>
<body>
<App>
<img src="1.jpg" alt="some">
</App>
</body>
/html>
```

Якщо при завантажені даного документа, він би був незалежний від JS, тоді дочекавшись завантаження стилів, сторінка стала би інтерактивною. В даному варіанті, сторінка не може стати інтерактивною, доки виконується скрипт. Тому, що компонент App створений за допомогою JS. Якщо час

виконання скрипту більше ніж 60 мс, час до досягнення інтерактивності буде залежить від всього часу, та часу на завантаження та компіляції скрипту. 130 КБ зжатого JS коду, розраховується приблизно в 1 МБ.

Для тесту візьмемо два ноутбуки та сайт з IT-новинами, під адресую habr.com. У якості веб браузеру буде використовуватись Google Chrome, наступної версії 87.0.4280.88. При роблені тесту, будемо відкривати даний сайт, и відстежувати його продуктивність за рахунок інструменту DevTools. Час заміру є 6 секунд. Це середній час завантаження сайту.

Перший ноутбук має наступні характеристики. Процесор на базі Intel Core i5 2410-M, 8GB оперативної пам'яті.

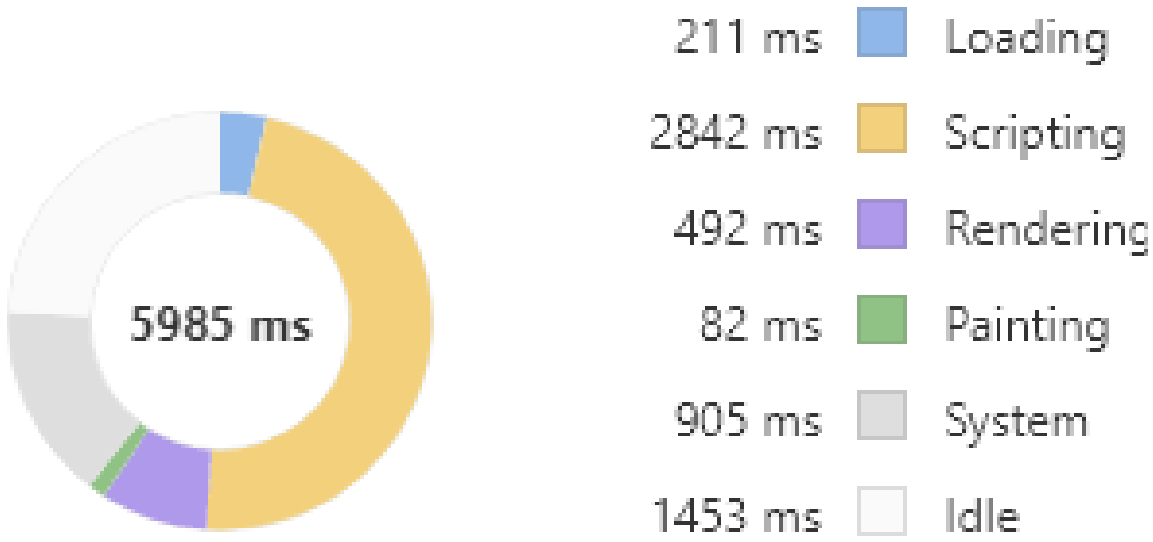

Рисунок 1.4 – Результат проведення тесту продуктивності на першому ноутбуку

При проведені тесту, маємо наступні результати. Швидкість рендерінгу сайту зіставила 492 ms.

Другий ноутбук має наступні характеристики. Процесор на базі Intel Core i3 6006б 8GB оперативної пам'яті Він показав гірші результати, при

однакових умовах. Швидкість рендерінгу сайту зіставила 1054 ms, що в два рази більша ніж у першого процесору.

Рисунок 1.5 – Результат проведення тесту продуктивності на другому ноутбуку

При виконані тесту, зрозуміло що швидкість рендерінгу прямо пропорційна швидкості праці CPU клієнта. Тому придбання, або апгрейд вашого процесору, або пристрою клієнта, може зменшити час чекання завантаження сторінок. Даний сайт використовує наступний стек технологій, PHP 5, фреймворк Propeller, MySQL. В даному випадку, буде умисно використати мінімізовані та легко важкі бібліотеки, для прискорення завантаження цього веб-додатку.

1.4. Робота з бібліотеками, та фреймворками

Розглядаючи веб-додатки які були написанні за минулий час, можна знайти багато різних бібліотек, які підключенні до додатку. Проаналізувавши їх, можна зробити висновки, потрібні лі вони вам, або їх все необхідно замінити на більш сучасні рішення.

Ряд причин, при яких задуматись о зміні бібліотек, або їх обновлення:

- відказ від підтримки бібліотеки, розробниками. Якщо розробки бібліотеки закінчили її підтримку у 2015 році, то потрібно задуматись о її зміні. Важливо обновлювати бібліотеки на стороні клієнту, тому що кожен день знаходять нові уразливості. За допомогою їх можуть вкрасти данні, або зламати сайт;
- бібліотека багато важить. Якщо ви використовуєте лише пару функцій з бібліотеки у веб-додатку, і для легкості написання коду підключили її, таке рішення є нераціональним, и може значно вплинути на продуктивність веб-додатку, або сайту.

Якщо бібліотека важка, і вона необхідна для існування додатку, є можливість відкласти загрузку бібліотеки, до тих пір, поки не завантажаться начальна сторінка веб сайту.

Заміна важких бібліотек, це важливий крок, до покращення продуктивності вашого веб додатку. Наприклад заміна бібліотек React та на SvelteJS [4], у маленькому додатку, значно прискорить завантаження. Тому, що SvelteJS, це дуже легка бібліотека у використанні. А React, важка, яка використовується для великих веб-додатків.

Одна з речей, якої потрібно підтримуватись наступна. При розробці нового додатку з нуля, важливо підійти до питання стеку проекту. Якщо ви вибрали неймовірно великий фреймворк, для розробки невеликого сайту, або веб додатку, то наслідки можуть призвести до втрати продуктивності програмного продукту.

Вибираючи стек, потрібно використовувати найбільш легке рішення, яке може вирішити вашу задачу, на проект. Наприклад, бібліотека React 15.6.1 важить 20.76 KB. На даний час, це дуже багато і тому буде раціональніше використати бібліотеку Preact [5], яка повністю є клоном React, і має вагу в 3 KB. Та фреймворк Next.JS, який дозволить створити серверну візуалізацію. Дане рішення дасть змогу зменшити вагу вашого веб-додатку, і збільшити його продуктивність.

Потрібно знати цілі, при виборі того, чи іншого, а не вибирати те, к чому звикли, якщо його вибір не є раціональним.

1.5. Веб-кешування

Кожна людина, будь то активний користувач інтернету, або більш консервативна людина, яка заходить на сайт з рецептами, або погодою, постійно використовує кешування, навіть недогодуючись про це, і як воно працює. Веб-розробники знають, що кешування спрощує створення продуктивних, та швидко дійних веб додатків, це дає змогу постійно проводити оптимізацію, перенавантажених серверів, від тисячі запитів.

Представимо, що в нас є веб-додаток, на якому не реалізовані методи кешування. Без кешування, сервер на якому розміщений додаток буде дуже залежний від свої обчислювальної потужності. Однією, з основної функції кешування є завантаження таких статичних ресурсів, як:

- статичні файли HTML;
- файли стилів, CSS;
- зображення веб додатку;
- JavaScript-файли.

Знаючи, що на кожен запит користувача, сервер повинен дати відповідь, але коли людина робить запит на завантаження сторінки, то крім цього відбуваються чотири окремі запити - по одному ,на кожну категорію вище.

Статичні файли сайтів, можуть бути об’ємними, і якщо велика кількість людей по всьому світу легко перенавантажать сервер, і всі сторінки на веб додатку будуть довго завантажуватись, що приведе до збільшення кількості відказів клієнтів, та зменшення конверсії що є недопустимим. Приклад такого доступу, на рисунку 1.6

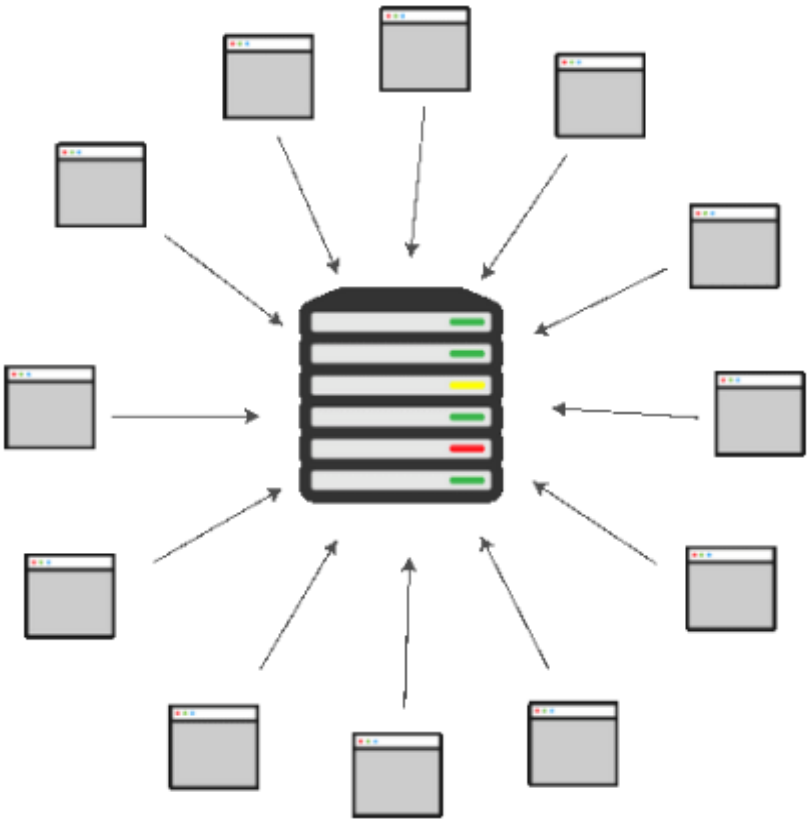

Рисунок 1.6 – Приклад звернення всіх клієнтів, до одного серверу

Щоб вирішити дану проблемо, можна взяти і купити декілька серверів, але це дуже дорога і нераціональна процедура. При такому підході, масштабування буде проблематичним і дорогим. В ідеалі, потрібно знизити навантаження на сервери, якимось методом, щоб зберегти відповіді на популярні запити. Серверу не потрібно буде обробляти кожен запит. Відповідь буде братись із кешу. [6]

1.5.1. Кешування на стороні сервера.

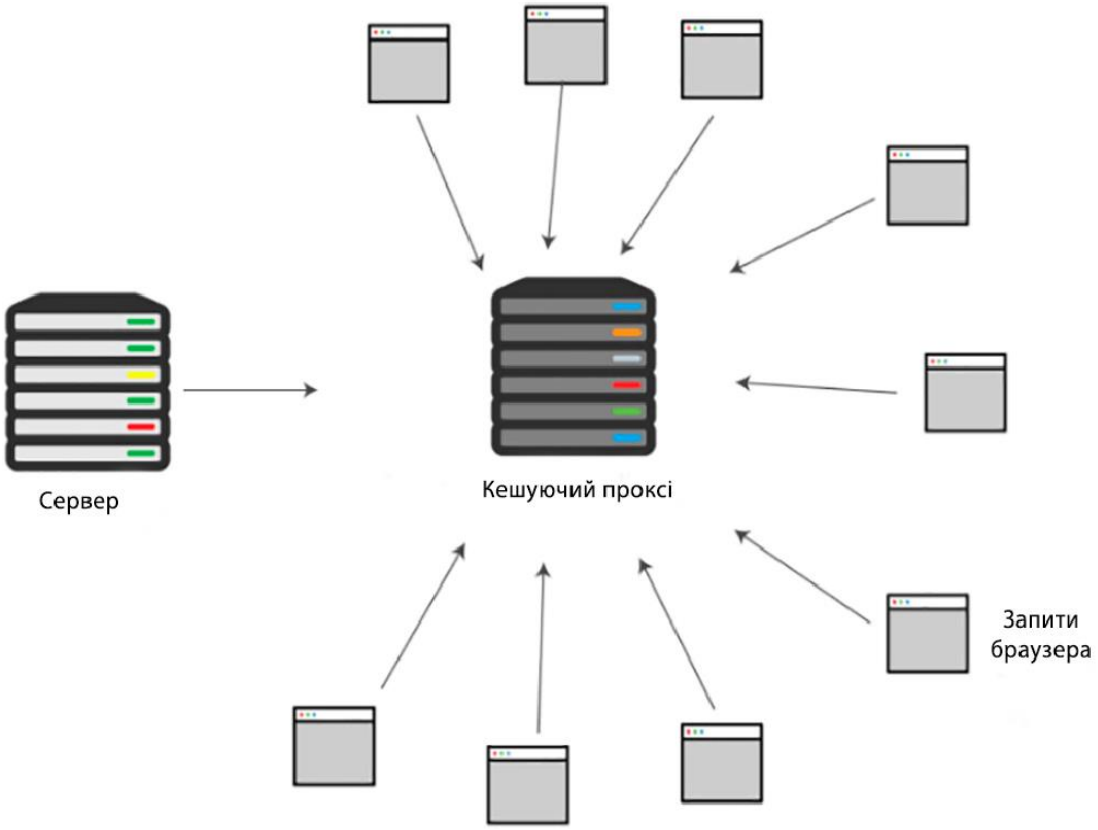

Рисунок 1.7 – Схема звернення клієнтів, с допомогою кешуючого серверу

На данному зображені, відображається керуючий проксі-сервер який зберігає в собі статичні файли, які були описані вище. Ці файли даний сервер використовує у якості відповіді на популярні запити. Він буквально перехвачує такі запити і дуже швидко дає на них відповідь. Це дозволяє знизити навантаження на основний веб-сервер, це можна розглядати як делегування обов’язків.

Політика кешування дуже важлива річ. На даному проксі серверу будуть зберігатись різній файлі, які кешовані у різний час. І дана політика буде вирішувати, потрібно лі ще зберігати данні файли, або ні. Це буде залежати від популярності запитів.

1.5.2. Кешування на стороні браузера

Даний метод кешування, унікальний тому, що він використовується для кожного окремого користувача. В настройках браузера, можна виставити свої параметри кешування, які підходять для певних цілей. В даному кеші, зберігаються всі файли і документи, які він завантажив по протоколу HTTP, методом GET. Після збереження даних, користувач буде мати доступ, к сторінкам які він завантажував раніше. Це дозволяє переміщатись при навігації, не втрачавати час на повторне завантаження сторінки і контенту.

При втрати зв'язку з інтернетом, можна буде зайти, на сторінки, які зберіглись в кеші. Розглянемо схему кешування на стороні браузера.

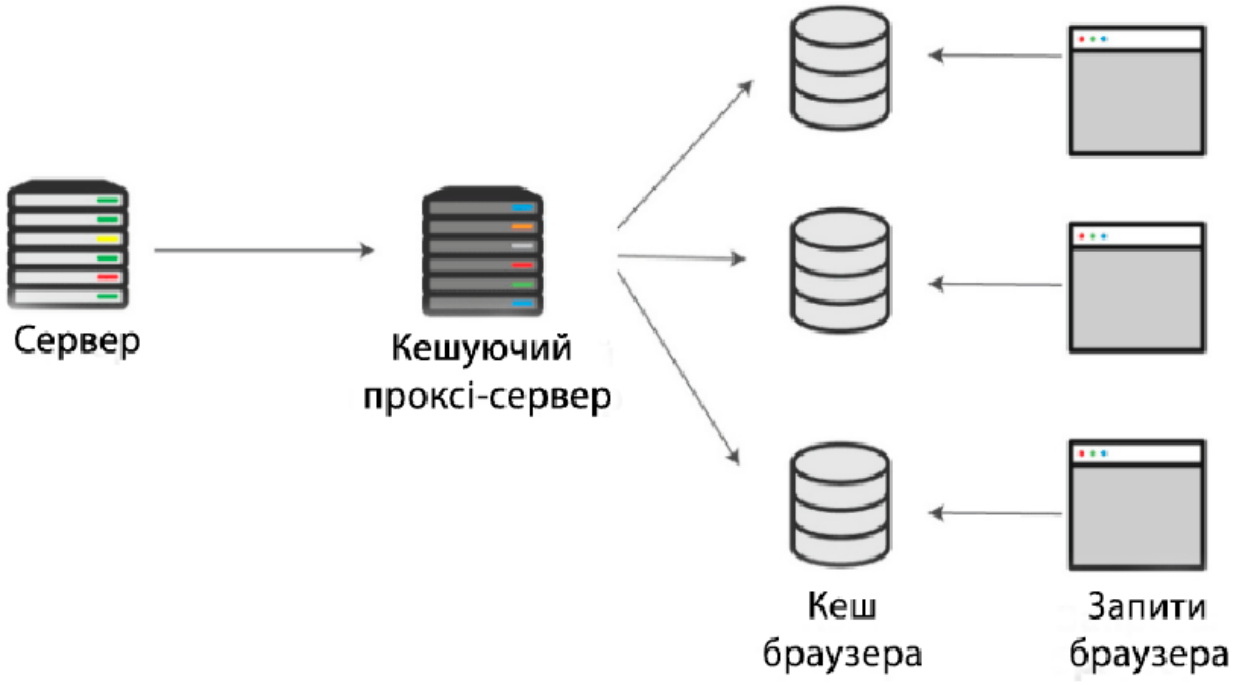

Рисунок 1.8 – Кешування на стороні браузера

Такий підхід дуже зручний для користувача, у випадку частого відвідування сайтів наприклад rozetka.com.ua, а також дозволяє цьому сайту економити на серверних витратах. Це відбувається за рахунок меншої кількості повторних запитів.

Важно зрозуміти, що файли у кешу браузера не з’являються самі по собі. Клієнту потрібно зробити початковий запит, та завантажити сторінку. Ресурс, який завантажив клієнт, в теорії може зберігатися вічно, але існує проблема, об’єм даних сховища кінцевий. Тому, у зв’язку к цим, дані періодично видаляються. Цей процес називається cache eviction.

Також існує проблема, коли користувач завантажив сторінку, вона зберіглась у кеші при цьому данні на сервері змінились. Щоб запобігти конфлікту, кеш потрібно оновлювати. У зв’язку з цим, сервера і кеш на стороні клієнта, встановлюють freshnessLifetime, термін дії, або придатності. Дані параметри записуються у заголовках.

Розглянемо приклад, завантажимо сторінку facebook.com.

При першому завантажені, даних на стороні клієнта не було, то перші данні можна важити початковими, та найбільш важкими. Їх можна побачити на рисунку 1.9.

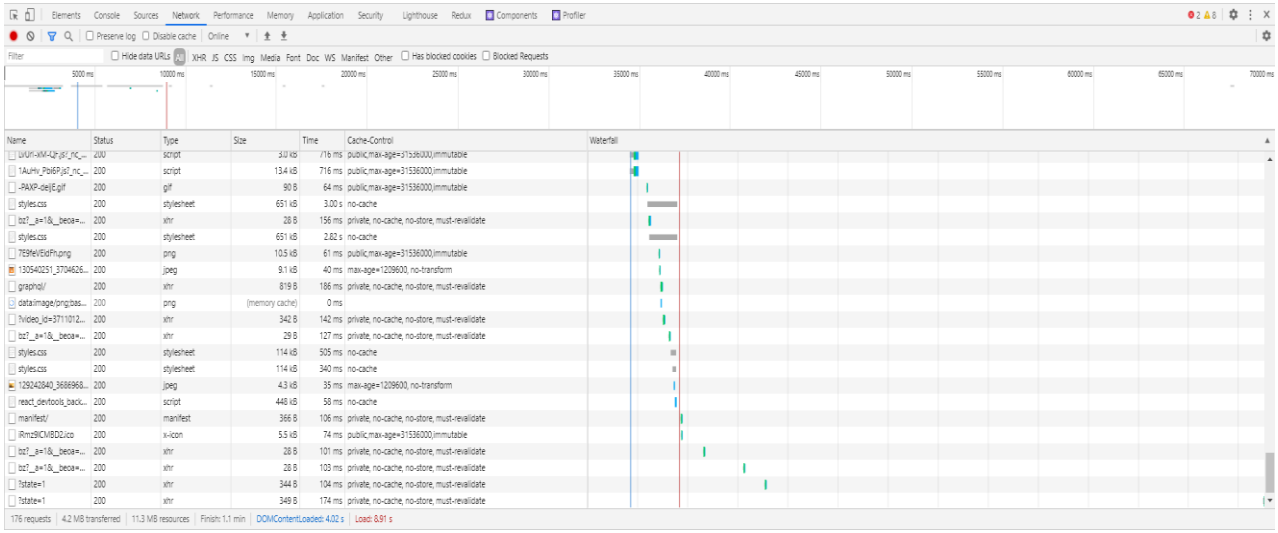

Рисунок 1.9 – Початкове завантаження даних, веб сторінки

Початкове завантаження сторінки, зіставило 8.91 секунди, а об’єм переданих даних 4.2 MB. Після цього перезавантажимо сторінку, щоб отримати результати, з використання кешування на стороні клієнту. Результати можна побачити на рисунку 1.10.

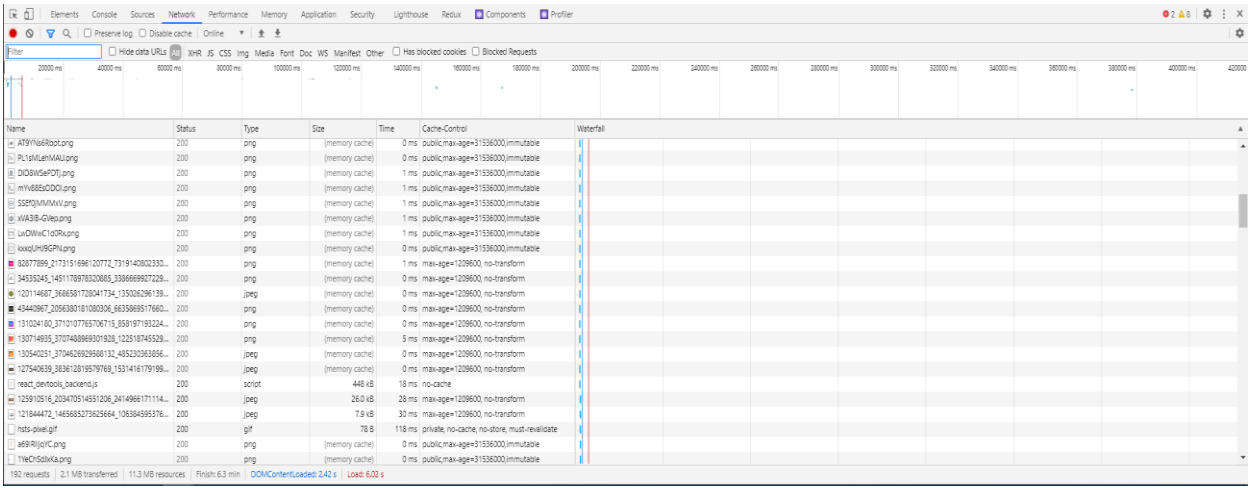

Рисунок 1.10 – Завантаження кешованої сторінки

При використані кешування, отримали наступні результати. Загальне завантаження сторінки, зменшилось на 2.89 секунди, та кількість переданих даних по мережі, на 2.1MB. Це ті дані, які закешувались у клієнта.

1.6. Мережі доставки контенту, CDN

Перш за все, CDN це мережа доставки контенту, це один або група серверів, розташованих в різних куточках світу, яка представляє веб-контент. Концепція даного методу в тому, щоб розмістити декілька точок, за мережу початкового серверу. Це дозволить веб-додаткам, більш швидко оброблювати запити клієнтів, що дасть користувачу більш приємний опит користування додатком. Розглянемо метод праці і особливості CDN [7].

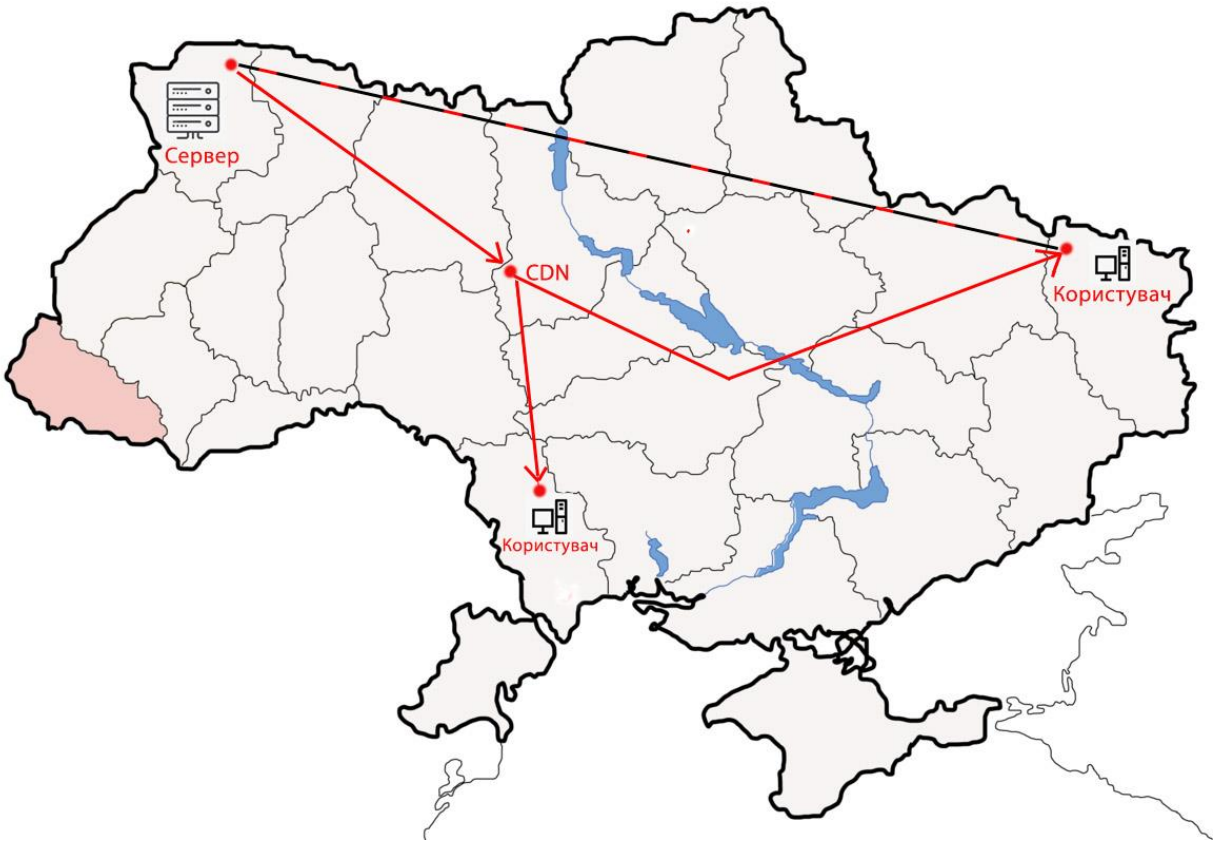

Рисунок 1.11 – Приклад роботи CDN, межах країни

Наприклад в нас, э веб-додаток, яким користуються люди, на всій території України. Основний сервер розволожений на заході країни, у місті Луцьк, це 350км від Києва. А користувачі знаходяться, в таких містах як, Одеса 608км від Луцьку, та Луганськ 1030км від Луцьку. Чим більша відстань між користувачем та сервером , тип більше час відповіді.

У даному випадку, користувач із Одеси, або Луганську переадресовується до ближнього кеш-серверу в якості CDN, завдяки цьому доставка статичного контенту відбувається більш швидше.

Був розглянутий випадок, де початковий сервер знаходить в одній країні, з користувачами. Але початковий сервер може знаходитись будь де, як правило в Германії, або США. І тому напряму звертатися до нього, з другої країни, буде витратною за часом справою.

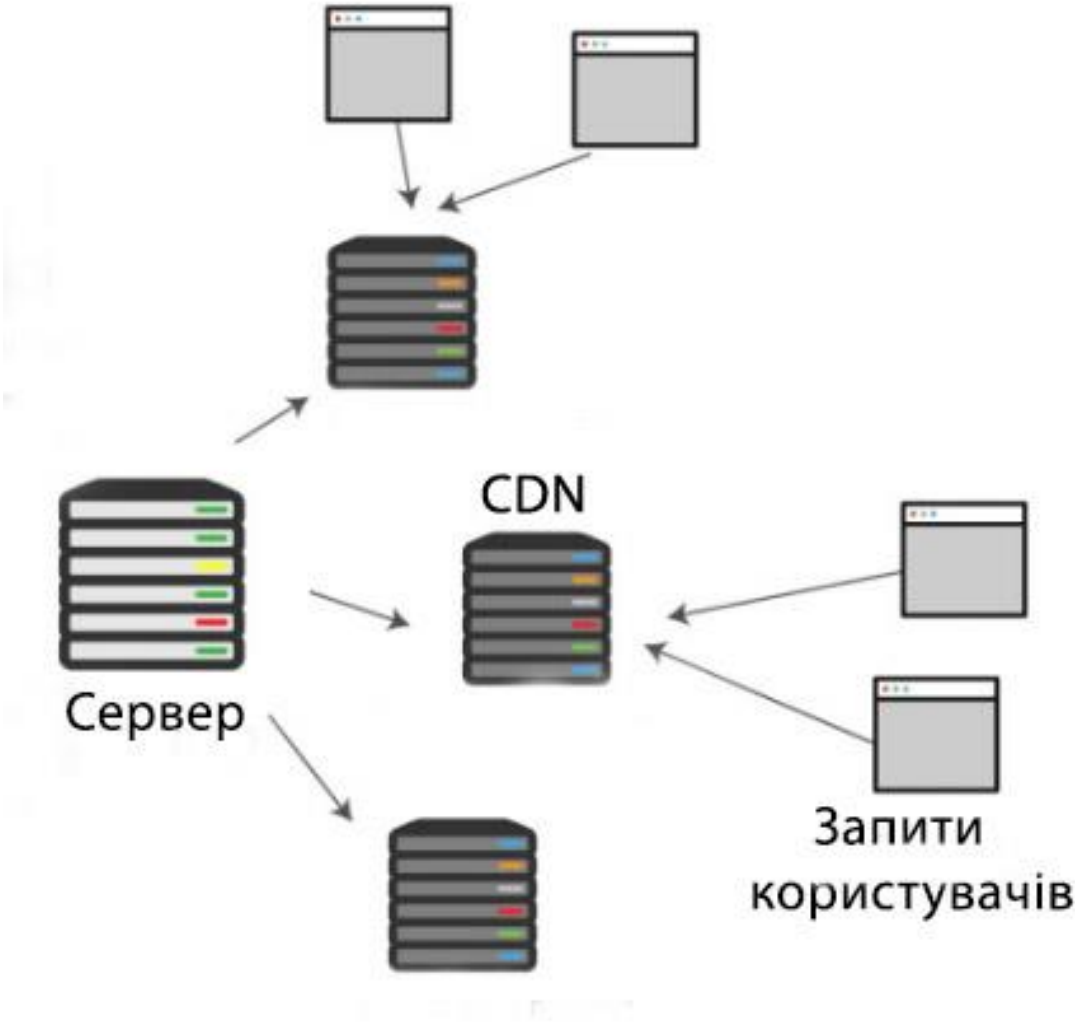

Рисунок 1.12 – Використання CDN, для запитів користувачів

Таким чином, початковий сервер відправляє копію статичних ресурсів на кожний проксі сервер в мережі CND, які все обробляють всі локальні запити на місцях, доки ресурси на них не застаріли. Також CDN дає широкий спектр можливостей, для прискорення сайту:

- установка заголовків клієнтського кешування. Дуже корисна функція, для прискорення веб додатку, проставлення заголовків кешування контенту браузером клієнта. Такий як, cache-control, та max-age для встановлення часу життя кешу;

- стискання текстових ресурсів. Це базова функція всіх CDN, які зменшують розмір початкових текстових ресурсів;

- оптимізація зображень. Зображення на веб додатку стискають за допомогою популярних форматів стискання, таких як: WebP. Оптимізація зображень буває двох типів, з втратою якості зображення, та без втрати;

- головною перевагою CDN, э низькі затримки. Це робиться за допомогою створення географічної архітектури мережі, з розташуванням хостів у точках густонаселених регіонів, столицях. Наприклад у 2016 році, Cloudflare розташував у Києві свій дата центр, завдяки цьому сайти з України будуть краще відзиватись. Найбільш корисно буде, якщо аудиторія веб-додатку розташована на відстані, більше 2 тисяч кілометрів. На прикладі України, якщо сайт призначений для українських користувачів, тоді є смисл розташувати головний сервер, в дата центрі Києва, щоб відстань до всіх крайніх точок України була плюс, мінус однакова;

- оптимізація контенту. Зміна контенту при доставки, проводиться мінімізація коду. Це дуже рисковий процес, і не всі поставники CDN представляють його.

1.7. Зменшення часу відповіді серверу, DNS.

Щоб переглянути якусь сторінку веб сайту, або веб додатку, клієнт робить GET запит на сервер. Та сервер дає йому відповідь, в якій лежить файл, маючий HTML. Ця система називається HTTP.

Але якщо використовувати звичайне ім’я сайту, то маршрутизатор не зрозуміє куди слати запит. Це незручно, тому, що користувачам потрібно буде запам’ятовувати IP-адрес, веб сайтів. Щоб вирішити цю проблему, зробили систему доменних імен(DNS). Ваш веб-браузер буде використовувати DNS для перетворення імені сайту, наприклад (habr.com) в IP адресу [8]. Цей процес, перетворення доменного імені, в адресу називається, ресолвингом(DNS-resolver), або дозволянням DNS.

Щоб зрозуміти як працює DNS, давайте розглянемо рекурсивний запит, в звичайному сценарії дозволу DNS, або ресолвингу:

Для прикладу візьмемо сайт www.habr.com. Розіб'ємо дану URL-адресу на частини.

www – subdomain

habr – second level domain

com - top-level domain, верхній рівень.

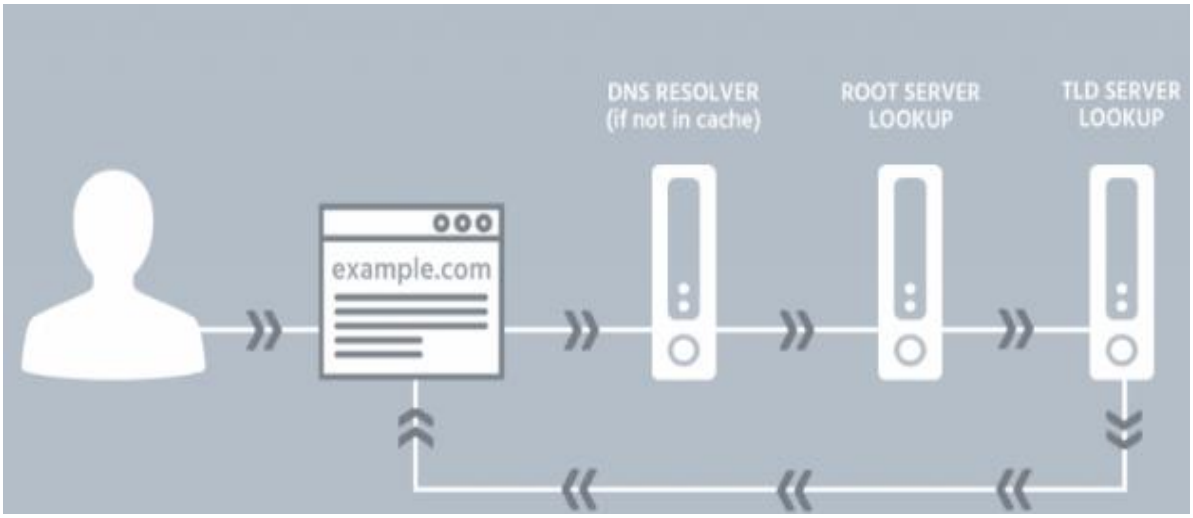

Рисунок 1.13 – Приклад запиту, к імені www.habr.com

- клієнт DNS, робить запит у свого рекурсивного перетворювача www.habr.com;

- ваш рекурсивний перетворювач робить запит у кореневого сервера імен. Він запитує у кореневого серверу, де найти йому необхідні відомості про адреса в доменній зоні верхнього рівня;

- кореневий сервер імен, направляє ваш рекурсивний перетворювач на сервер домену верненого рівня(TLD). У нашому випадку це є .com;

- ваш рекурсивний перетворювач запитує у повноважного сервера TLD .com адрес www.habr.com. Це значить, що TLD сервер питає у www.habr.com його IP-адресу;

- рекурсивний сервер клієнту запитує у авторитетного серверу, та отримує у відповіді IP-адресу для www.habr.com , 134.135.136.223;

- ваш рекурсивний перетворювач кешує відповідь в протягом часу життя (TTL), який указаний в запису, и повертає его вам.

Приклад запиту, на www.habr.com.

```
ad@moodle:~$ dig www.habr.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.habr.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39739
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.habr.com.                  IN      A

;; ANSWER SECTION:
www.habr.com.           2133    IN      CNAME   habr.com.
habr.com.               2132    IN      A       178.248.237.68

;; Query time: 44 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sun Oct 25 22:28:24 UTC 2020
;; MSG SIZE  rcvd: 71
```

Рисунок 1.14 – Приклад запиту, к імені www.habr.com

За допомогою інструменту dig [9], зробимо запит даний сайт. Після виконання запиту, отримаєм наступну відповідь. Розглянемо основні пункти.

Header - в цій частині, будуть знаходитись технічні дані, об відповіді, який отримали від запитуваного органу. Також ми маємо id операції, її статус, в даному випадку статус NOERROR. Це означає, що запитуваний орган обслуговував запит без помилок.

ANSWER SECTION – це основний розділ, на який потрібно звернути увагу.

По умовчанню, даний інструмент запрошує запит A. Це означає, що він запитає список всіх адрес, даного домену.

Наступний розділ починається з строки Query time. Цей розділ виводу dig, показує користувачу всю статистку запиту. Час запиту, сервер я з якого відбувався запит, у даному випадку (localhost) і дату запиту.

Швидкодія даного запиту залежить від швидкості DNS-провайдеру. Щоб максимально збільшити швидкість запиту, потрібно використовувати найшвидші DNS. Цю статику можна отримувати кожен місяць, за допомогою SolveDNS [10].

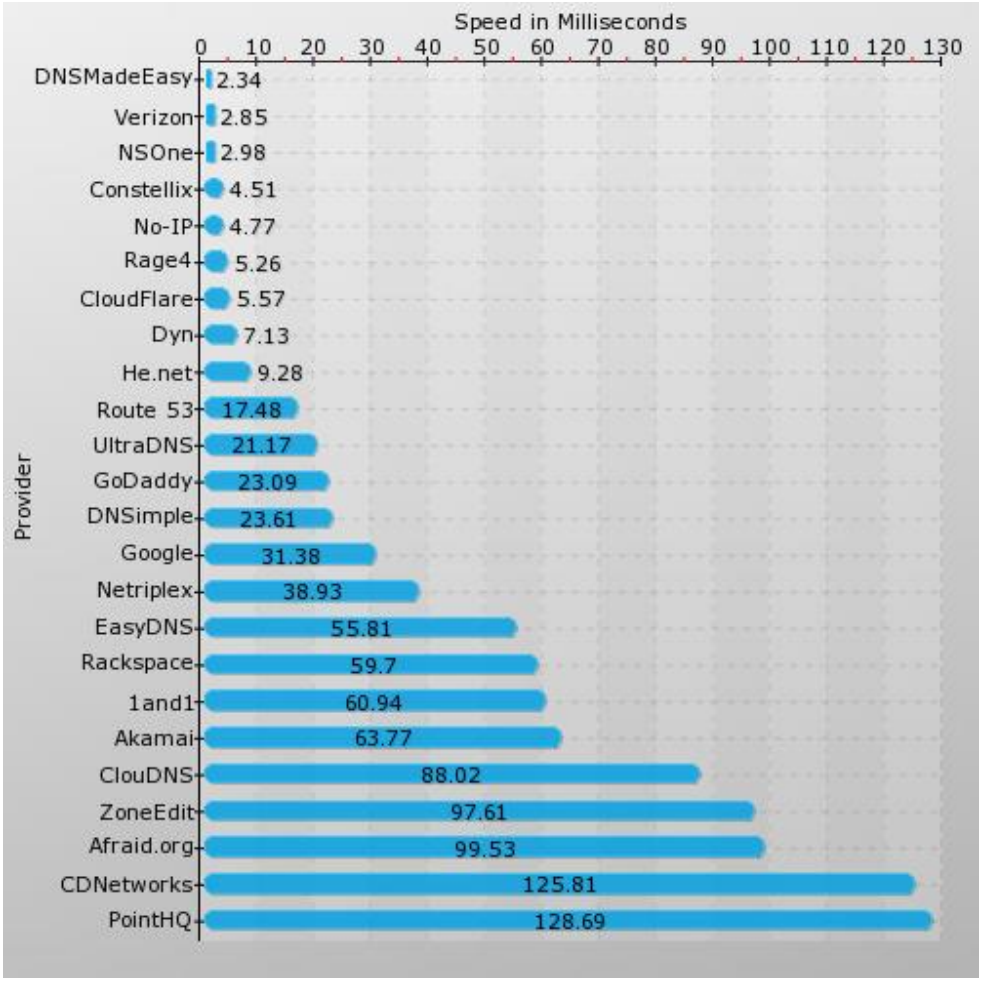

Рисунок 1.15 – Зрівняння швидкостей DNS-серверів.

1.7.1. Зменшення кількості запитів DNS.

DNS встановлює відповідність хост-імен к їх IP-адресам. Можна привести приклад, це працює, як телефонний довідник дозволяє узнати номер телефону людини, по його імені.

При роблені запиту на , www.habr.com перетворювач DNS повертав його адресу. процес, може займати від 30 до 150мс, щоб його виповнити.

Ваш Браузер буде чекати завершення цього DNS-запиту, а вже після цього почне завантажувати контент.

Зменшення кількості унікальних імен хосту, потенційно може зменшити кількість паралельних завантажень контенту. Наприклад, зображення, які має веб-хост (habr.com) знаходяться на 4 різних хостах. З цим ви забезпечите 4 паралельних завантажень, з цих хостів.

Зменшення кількості DNS-запитів, можуть значно зменшити час завантаження вашого веб-додатку. Важливо розділити завантаженні

1.8. Уточнена постановка задачі

Таким чином, взявши до уваги всі приведені методи, можна сказати, що програмні методи прискорення роботи веб-додатків найбільш ефективні. Тому у даному досліджені буде розглянутий програмний метод.

Враховуючи це, можна сформувати наступну уточнену задачу для дослідження. Зважаючи на те , що швидкодія стала головним, та самим важливим фактором, який визначає зручності використання інтернету в цілому, і окремих веб-додатків, у даній роботі буде проведене дослідження над векторним методом роботи з картинками. Тому , що зображення мають значний вплив на веб додатки.

Важливим елементом, буде дослідження зміни завантаження роботи сайту при використані сучасного методу роботи з векторними зображеннями. Зрівняння даного векторного методу, з схожими методами праці, такими як Lazy Load, та заміна зображення на зображення меншої якості.

1.9. Висновки до розділу

Таким чином, у даному розділі розглянуті основні програмні, та мережеві методи для збільшення продуктивності, та зменшення часу завантаження веб додатку. Кожен метод заслуговує своєї уваги, на його

дослідження. При їх розгляді, було визначено, що найбільший вплив на продуктивність та швидкодію завантаження впливає розмір JavaScript файлів, та кількість і вага картинок, які завантажується.

Будуть проведені дослідження на вплив зображень на початкове завантаження сторінки. Розглянувши особливості вибору зображення, було зрозуміло що SVG формат э дуже ефективним. Методи роботи з зображеннями, такі як Lazy Load, або заміна на більш гіршу якість використовуються у даний час, але вони мають багато недоліків. Тому на основі зібраних даних можна перейти до наступних досліджень впливу зображень.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ МЕТОДІВ ДЛЯ ЗБІЛЬШЕНЯ ПРОДУКТИВНОСТІ ВЕБ-ДОДАТКІВ

2.1. Вплив часу завантаження веб-додатків на користувача

Щоденно звичайний користувач, проглядає дуже багато різних сайтів, від онлайн магазинів до соціальних мереж. Кількість інформації, та реклами дуже велика.

По даним дослідження Akamai [11], якщо ресурс , який ви маєте завантажується довше ніж 3 секунди, тоді ви втрачаєте половину потенційних відвідувачів сайту. На сьогоднішній день користувачу більш важливо, швидкість завантаження веб-додатку, ніж то як він буде виглядати, або якість новомодні функції, які реалізовані на сайті. Чим більша швидкість вашого сайту, тим більше користувачів ви отримаєте. Оптимізувавши цей параметр, дозволить покращити опит користувача, та збільшити конверсію і дохід.

Розглядаючи дані дослідження, можна зрозуміти що 79% користувачів, у яких зашились негативні емоції при використанні сайту, з меншою ймовірністю зроблять покупку на цьому сайті знову. Як приклад можна розглядати інтернет магазин Djobang.

Також потрібно звернути увагу на мобільних користувачів, тому що їх більшість. Результати дослідження кажуть, що 64% користувачів, чекають що сторінки веб сайту, будуть завантажуватися не більше 4 секунд.

Якщо розглядати залежність швидкості завантаження сайту, до прибутку, то можна сказати, що зменшення завантаження сторінки на 1 секунду, може принести інтернет магазину до 7% збільшення прибутку. Наприклад, якщо ваш ресурс, приносить 100к грн в день, то ці показники збільшяться до 107к.

Так як сучасні пошукові системи, відстежують швидкість завантаження сайтів, цей показник є дуже важливий. Він впливає на показ сайту у системі. Тому важливо максимально зменшити цей показник , за допомогою програмних, апаратних та мережевих методів збільшення продуктивності роботи веб додатку. Затримка швидкості завантаження сайту на 1 секунду, приведе до скорочення показів сторінок на 11%.

2.1.1. Розрахунок та аналіз показника відмов

Показник відмов – це процент користувачів, які зайшли на сайт, але протягом 1-10 секунд його покинули, прямо з головної сторінки сайту. Даний показник можна визначити за формулою:

$$Rb = Nv / Ne , \qquad (2.1)$$

де R b- це показник відмов;

Nv – кількість людей, які переглянули одну сторінку, на веб-додатку;

Ne – загальна кількість переглядів сторінок.

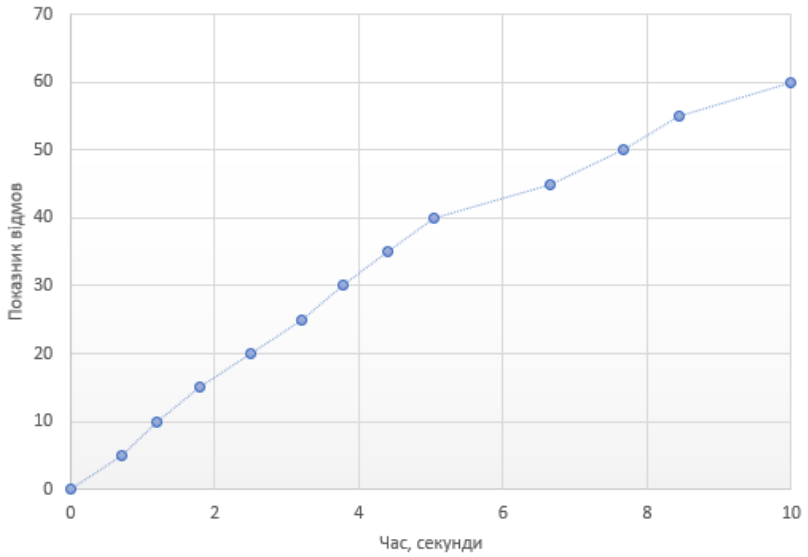

Рисунок 2.1 – Приклад залежності швидкості завантаження сайту, до показника відмов

Перевірити наявність нав’язливої реклами, це можуть бути банери, які займають багато екранного місця, і їх кількість більша ніж 1.

Швидкість завантаження веб-додатку, довготривале завантаження сторінок дає негативний опит в користуванні сайтом, і відштовхує користувачів.

Проаналізувати контент на сайті, та покращити якість трафіку, для залучення потенційних клієнтів.

Обов’язково мати мобільну версію вашого веб-додатку.

Проаналізувати функціонал, тексти та дизайн на веб-додатку. Якщо текст тяжкий, потрібно переписати його. Зробити зміну вашого дизайну, якщо він все морально застарів.

Проаналізувати навігацію на сайті, зробити її максимально зручною.

Використавши популярні сервіси, проскакувати свій додаток, на більш розповсюджені проблеми, і вирішити їх.

Середній показник відказів, це процент який демонструє певну тенденцію по веб-додатку. На цей показник впливає дуже багато факторів. В цілому, існує така норма показника відказів:

Більше 80%, дуже високий показник який вказує на дуже погану ефективність сайту, що потребує негайних змін.

70-80%, високий показник, все ще потребує роботи над веб-додатком, для його оптимізації.

60-70%, середній показник всіх веб-ресурсів.

Менше 60%, потрібно стримитись, для досягнення таких показників.

Щоб визначити, точну кількість відказів, можна підключити Google Analytics, яка необхідна для аналізу. Нема кордонів для досконалості, тому потрібно завжди проводити роботи, для покращення оптимізації на сайті.

2.2. Метод бюджету продуктивності

Щоб запобігти втраті продуктивності були введені бюджети продуктивності. Бюджет продуктивності – це ліміт для попередження регресії. Бюджети продуктивності встановлюють певні стандарти, для швидкодії вашого сайту, або веб додатку.

Маючи інтернет магазин, або якийсь інший сервіс, ви можете встановити певний рівень продуктивності для вашого сервісу.

Існує багато різних способів, зробити бюджет швидкодії. Їх можна визначити за такими параметрами:

- за часом, наприклад, взаємодія в мережі 4g повинна буди менше 3 секунд;

- за кількістю ресурсів, наприклад маючи JavaScript файл на сторінці, менше ніж 150кб;

- або за оцінкою спеціального сервісу, який визначає продуктивність вашого веб додатку. Наприклад Ligthouse.

Цей підхід буде корисний, для кожного веб сайту. Завдяки ньому, можна тримати певну планку, продуктивності вашого веб-додатку. Це зменшить ризики покидання своєчасного сайту.

Багато розробників забувають, що більшість користувачів, мають пристрої, які коштують 200$. Загально власники, при розробці PWA, вважають основною метою охоплення більшої маси нових користувачів. Далі розробники веб-додатків, використовуються різні інструменти для досягнення цієї цілі. При результаті, частіше всього готовий веб-додаток,

потребує рефакторінгу щоб забезпечити мінімальну продуктивність роботи. Це трапляється, тому що при початковій розробці сайту, тяжко визначити проблему, поки не буде занадто пізно.

Щоб уникнути таких проблем, при розробці, потрібні бюджети продуктивності.

При початку розробки, команді потрібно:

- на самому початку, встановити бюджети продуктивності;

- при масштабуванні додатку, змінювати бюджети звертаючи увагу на параметри мережі і пристроїв використання, які застосовуються на ринку;

- використовувати інструменти, які будуть допомагати відслідковувати прогрес, та запобігти регресу;

- використовувати технології, які максимально підходять для покриття необхідних задач, та забезпечення достатнього рівня продуктивності.

Бюджети продуктивності встановлюють певні кордони для визначення змін, при розробки веб-додатків. Ті зміни в кодовій базі, стануть кроком вперед, а які відштовхнуть користувача, та потребують негайних змін. Без цього любий веб-додаток спіткає крах.

Тому, у подальшому розгляді, за основу буде взяті бюджети продуктивності для наступних параметрів:

- час завантаження сторінки, 10 секунд;

- кількість малюнків, 20;

- загальна вага малюнків, 5 мб;

- розмір скриптів, 2мб.

2.3. Метод розрахунку інтерактивності, для різних показників

Час до першої взаємодії користувача, з сайтом це головний індикатор зручності сайту, по відношенню до користувача. Данну дії називають TTI, або час першої взаємодії.

Разом з зменшенням часу TTI, зменшується сприймає мий час завантаження сторінки, зо позитивно відображається на опиті використання клієнта, або користувача.

Щоб зменшити даний час, існує декілька способів це зробити:

- зменшити розміри картинок;
- провести мінімізація коду.

Для розрахування даного показника, потрібно використати наступну формулу, враховуючи бюджети продуктивності. Наприклад є заданні бюджети продуктивності для картинок, їх розміру. Формула для розрахунку TTI буде так:

$$TTI = (IMG_{KB}, Non - IMG_{KB}), \quad (2.2)$$

де IMG_{KB}, Non – встановлений розмір зображень;

IMG_{KB} – фактично-отриманий результат розміру зображень, при першій взаємодії з сторінкою.

При розгляді других статичних файлів сайту, розрахунок TTI змінюється. Якщо за основу, у бюджетах продуктивності взяли JS скрипти, то розрахунок даного показника буде наступним

$$TTI = f * (JS_{KB}, Non - JS_{KB}), \quad (2.3)$$

де JS_{KB}, Non – встановлений розмір зображень;

f – відображення типу , функція;

IMG_{KB} – фактично-отриманий результат розміру зображень, при першій взаємодії з сторінкою.

Для прикладу, візьмемо сайт habr.com. На його прикладі можна розрахувати час, перше ніж користувач зможе взаємодіяти з сторінкою. В даному випадку, повне завантаження сайту відбулось за 8.2. Час до першої інтерактивності зіставив 5.1s. Це можна побачити на рисунку 2.2.

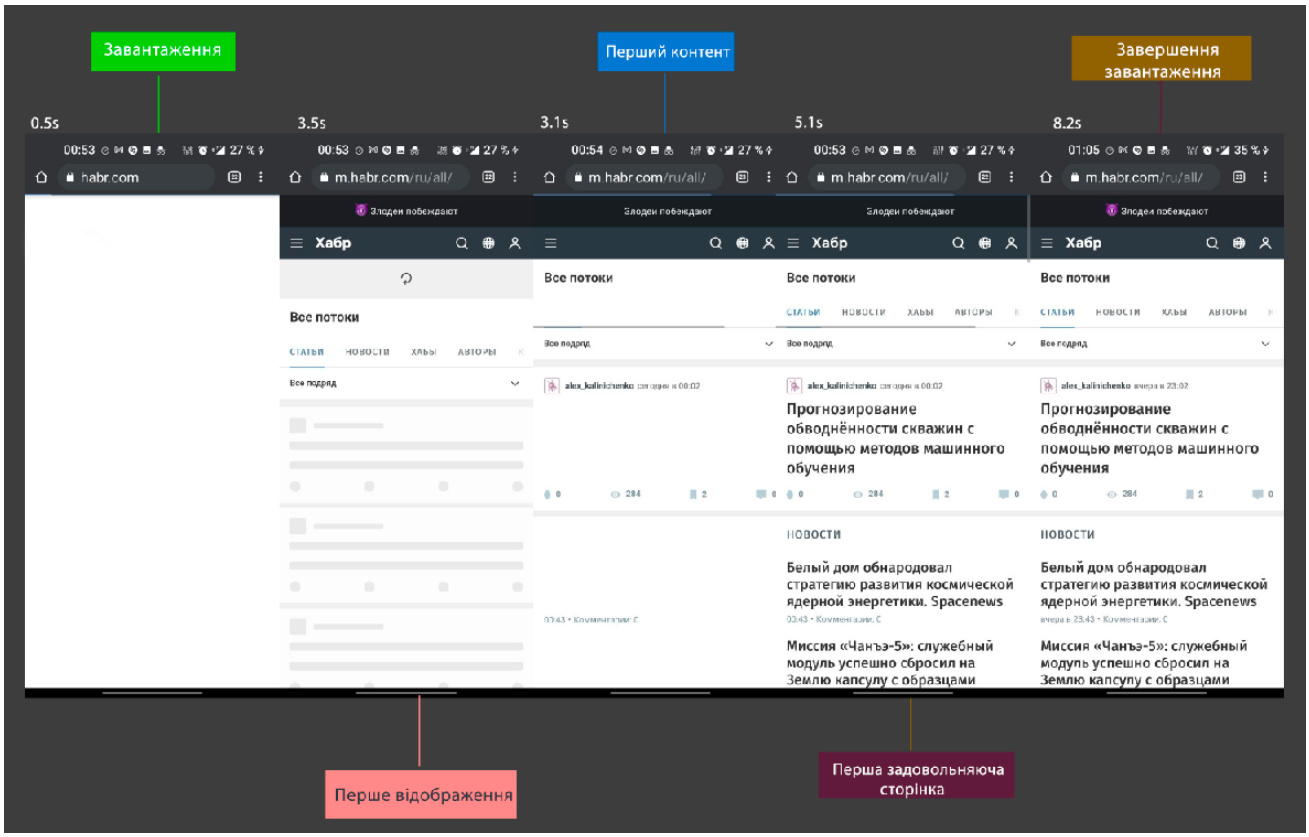

Рисунок 2.2 – Приклад аналізу інтерактивності сайту.

Так як даний параметр залежить від першого відображення контенту, можна сказати, що зображення та їх розмір значно впливають на цей параметр. Тому необхідно розробити метод праці з зображеннями, завдяки якому зменшиться початковий час завантаження.

2.4. Аналіз методу заміни зображення

При розробці нового методу праці з зображеннями, потрібно розглянути багато параметрів. Принцип дії методу, чи буде він працювати за допомогою JavaScript. Щоб це зрозуміти, можна розглянути існуючий метод праці з зображеннями, це заміна зображення. При аналізі цього методу можна виділити його алгоритм праці:

- завантаження сторінки;
- завантаження, та відображення зображення заглушки;
- фонове завантаження зображення високої якості;

- заміна зображення заглушки, на оригінал;

Розглянемо типовий тег img:

```
<img src="picture.jpg" alt="some picture"/>
```

Із основних атрибутів, src є головним, це маршрут до зображення, та alt, альтернативній текст, який відображається при довгому завантаження картинки. Існують також і інші атрибути тегу img, які відповідають за його параметри, висоту ширину місце знаходження, тощо.

Після аналізу, методу своєчасного завантаження картинки, можна виділити наступні його переваги, зменшення часу завантаження картинок, це збільшить початкове завантаження сторінки, що дуже важливо, легкість реалізації.

Недоліками цього методу, є підключення сторонньої бібліотеки jQuery, для його реалізації. Це вже не сама потрібна бібліотека, у 2020 році, і підключення її у проект на якому вже є бібліотека, не є раціональної дією. Існують варіанти з використанням чистого JS, або інших бібліотек розробки.

Тому є необхідність розробки варіанту, який не буде використовувати язик програмування, для своїх цілей. Використання векторного методу, я один з даних рішень, яке можна запропонувати.

2.5. Векторний метод роботи з зображеннями

Перша дія, це відмова від JS. Дану проблему можна було б вирішити на рівні HTML, і тегів. При внесені даної зміни зникне необхідність в використані язику програмування, для цієї задачі. Що в перевазі дасть менший час завантаження сторінки. За рахунок необхідності використання JS, та прискорить інтерактивність з користувачем.

Це, добавлення нового атрибуту, для тегу img. Розглянемо приклад:

```
<img src="picture.jpg" alt="some picture" pre=”picture.svg”/>
```

У даному випадку, з’явився поки що, абстрактний атрибут pre. Принцип дії, полягає в тому, що на рівні HTML, буде відбуватись перевірка даного атрибуту. Якщо він є, то спочатку на сайті буд завантажуватись SVG зображення, яке буде відображати суть картинки.

Функція даного атрибуту, буде закладатись в тому, щоб кодер зміг добавити, відносний або абсолютний маршрут для SVG формату зображення, яке на час завантаження картинок замінне їх. Так як SVG формат більш легко вісний, ніж PNG, JPEG. Це дасть змогу зменшити початкову вагу начальної сторінки, за рахунок заміни формату зображень. Після того як початкова сторінка завантажиться, замінити дані картинки, на оригінальні формати більш високої якості.

Цей метод, дасть дуже велике збільшення швидкодії першого завантаження сторінки. Першого, тому що після завантаження сторінка кешується, і повторне завантаження не займе багато часу. Це дасть змогу, може зменшити необхідність використання, такого методу для роботи з зображеннями як Lazy Load. Тому що для її використання потрібно підключати бібліотеку для її роботи, що вплине на загальний розмір веб-додатку.

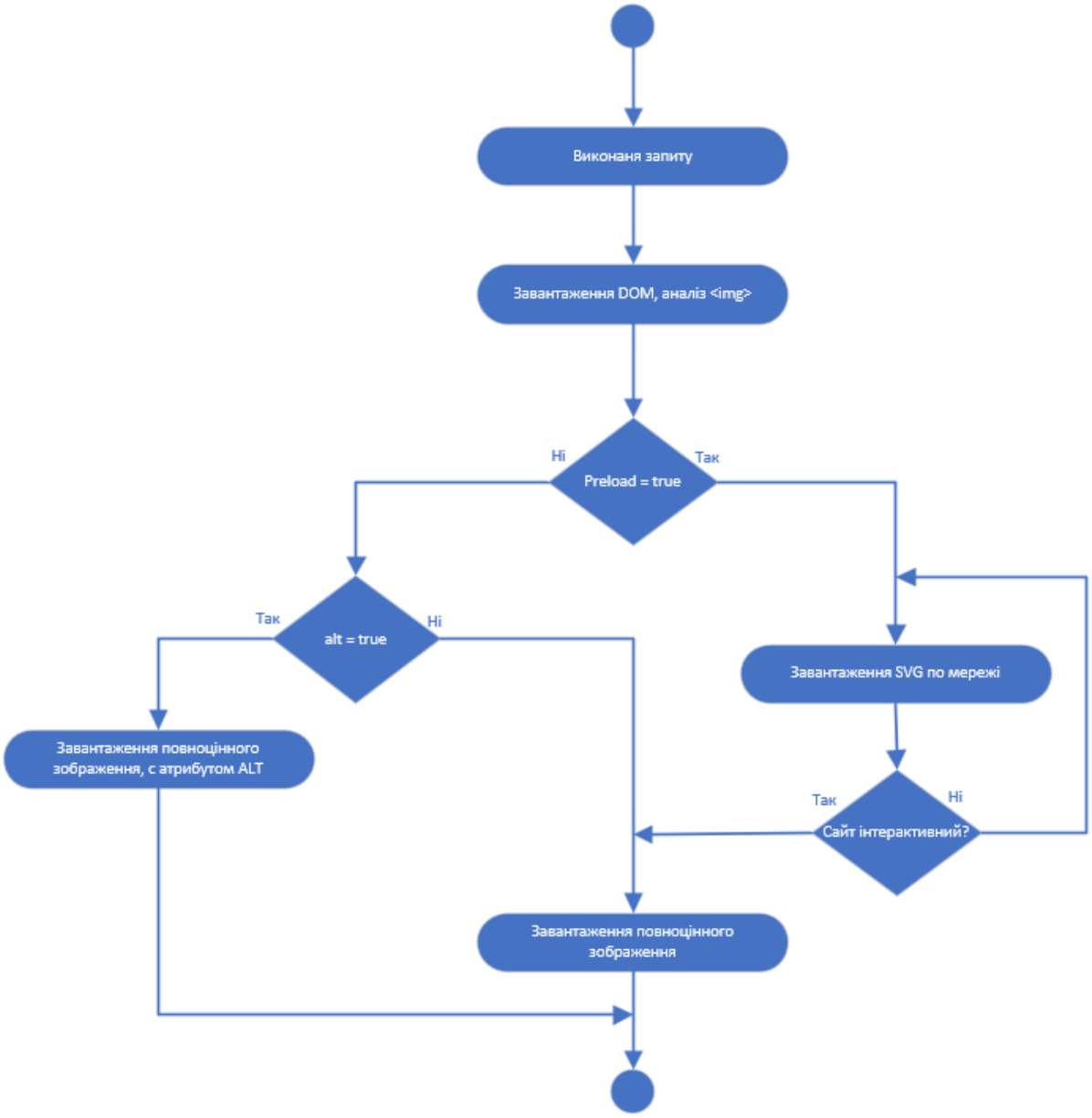

Рисунок 2.3 – UML-діаграма діяльності процесу завантаження зображення.

Для розрахунку його ефективності, можна використати наступну формулу.

$$e = (N_1 / t_1), \qquad (2.4)$$

де N1 – кількість зображень;

t1 – час повного завантаження зображень;

2.6. Висновки до розділу

У даному розділі були досліджені методи для збільшення продуктивності веб-додатків. Встановив бюджети продуктивності, можна задати певний рівень продуктивності. Із основних параметрів, можна виділити, Розмір зображення та кількість. Аналогічно й до мови програмування JavaScript.

Розглянувши вплив часу завантаження веб-додактів на користувача, встановили середній бажаний час, який становить 4 секунди. При збільшені норми, збільшується показник відмов на сайті.

Звернувши увагу, що зображення є невід'ємною частиною кожного веб сайту, є необхідність для створення нового методу роботи з ними, на основі використання векторної графікі. У наступному розділі буде досліджений метод, векторного зображення, його можливості, для ви рішення проблеми завантаження сайтів, та зменшення TTI.

Також у розділі були розглянуті основні методи, для розрахунку, ефективності, інтерактивності та відмов на веб-додатку.

РОЗДІЛ 3 ДОСЛІДЖЕННЯ ШВИДКОСТІ ЗАВАНТАЖЕННЯ

3.1. Дослідження продуктивності, на основі його бюджету

Спочатку, потрібно встановити бюджети продуктивності для вашого веб додатку.

Наприклад, візьмемо сайт, у якого встановленні наступні бюджети. Їх встановили у розділі 2

Дані бюджети можна записати у форматі JSON, для більш зручної роботи з ним. Приклад розробленого JSON. Даний файл встановлює певні бюджети продуктивності, для двох типів: скриптів вашого веб додатку, та картинок.

```
“timeToLoad” : [
{
        “value” : 10
},
“resourceSize” :   [
{
        “resourceType” : “script”,
        “value” :  150
},
{
        “resourceType” : “image”,
        “value” :  300
}
…
],
“requestAmount” : [
{
```

resourceType” : “script”,
“value” : 8
},
{
resourceType” : “image”,
“value” : 4
}
…
]

Принцип праці даної бібліотеки, буде такий. Користувач, або команда заповнюють бюджети продуктивності. Після цього дана бібліотека повинна відслідковувати ці данні. Якщо в процесі розробки, або на етапі впровадження у веб-додаток даної бібліотеки, ваші бюджети продуктивності виходять за рамки, то дана бібліотека оповістить відділ, який відповідаю за оптимізацію сайта.

Після цього команда розробників, зможе зробити поправки у код, якщо це був Alert по відношенню до JavaScript файлів, CSS або HTML. Якщо притримуватись даних бюджетів, та з самого початку розробки, притримуватись певних стандартів, то в майбутньому ваш веб-додаток буде оптимізован.

В даній таблиці, показані загальні данні. Наприклад, у веб додатку використовується 10 запитів, а встановлена норма 8. При такому розкладі, дана бібліотека сповістить розробника, щоб він оптимізував, та виправив дану проблему.

Таблиця 3.1 – Приклад бюджетів продуктивності веб-додатку

Тип ресурсу	Запити	Розмір файлу	Переваження Запитів	Переваження Файлів
General	32	958 KB	-	-
Загальні файли	12	431 KB	5 запитів	-
Скрипти	10	2.2 MB	2 запитів	0.2 MB
Малюнки	20	6 MB	-	1 MB
Стилі	2	33 KB	-	-
Шрифт	2	14 KB	-	-
Документ (HTML)	2	20 KB	-	-
Час завантаження	10с	-	-	-

Може статись ситуація, коли вже є готовий продукт, з недуже гарними показниками. І цьому ресурсу потрібно бути встановити бюджет продуктивності. Щоб це зробити потрібно досконало проаналізувати свій веб додаток. Перегляну які таймінги завантаження мають сторінки, стилі, JavaScript код або картинки має ваш веб сайт.

Проаналізувавши результати, можна буде встановити бюджети, враховуючи наді.

Одною, з головних функцій даної бібліотеки, буде аналіз даних та визначення часу завантаження веб-додатку, при зміні поточних бюджетів продуктивності.

Приклад моделі аналізу, можна розглянути на рисунку 3.1. За основу був взятий встановлений бюджет JavaScript файлу, із таблиці 3.1. На даному малюнку показана зміна, та час завантаження вашого веб додатку, при різних показниках бюджету.

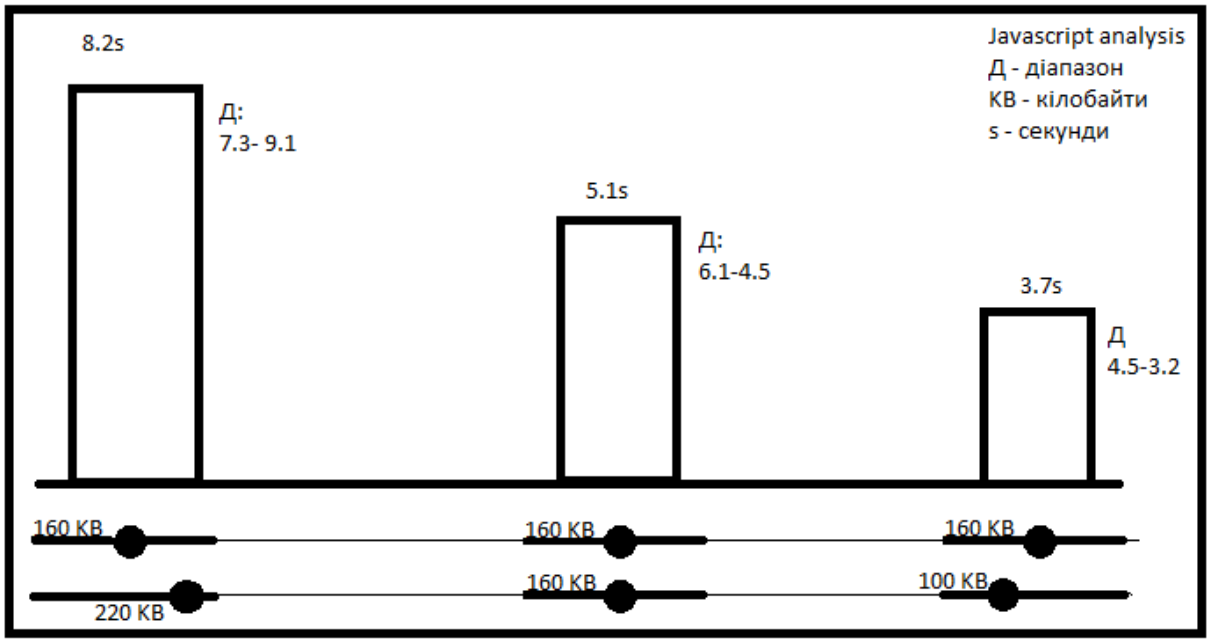

Рисунок 3.1 – Приклад аналізу продуктивності, при зміні JavaScript файлу

3.2. Моделювання методу, для своєчасного стискання зображення

Принцип дії даного методу, залучається в тому, що ми використовуємо аналогічне зображення, з знатна меншою вагою. Для реалізації даного методу, сервер повинен зберігати дві копії однакового зображення. Це не є недоліком, тому що вага копій, будуть важити навіть не кілобайти, а байти.

Після відкриття сторінки, клієнтом, замість основних важких картинок, завантажується картинка заглушка. В цей час, асинхронно завантажується оригінальне зображення, яке після завантаження замінить зображення низької якості.

Даний метод дасть користувачу розуміння, що на сайті щось відбувається. Клієнт зможе раніше побачити зображення, яке повинно завантажитись, тільки в гіршій якості, замість атрибуту “alt”.

Розглянемо класичне завантаження картинки.

```
<img align="center" src="picture.jpg" alt="some picture"/>
```

У данному випадку, якщо картинка має достатню вагу, то користувач замість неї, подаче текст, який описаний у атрибуту alt. Це все застарілий метод, для інтерактивності з користувачем.

Змоделюємо ситуацію. В нас є веб-додаток у якого, головне зображення має вагу 5Мб. Для людей, які мають підключений 4g, DOM завантажиться за 2ms ,а фонове зображення за 800ms/ Це дуже непоганий результат, але зона покриття 4g не досконала, тому в цих місцях буде працювати 3G інтернет.

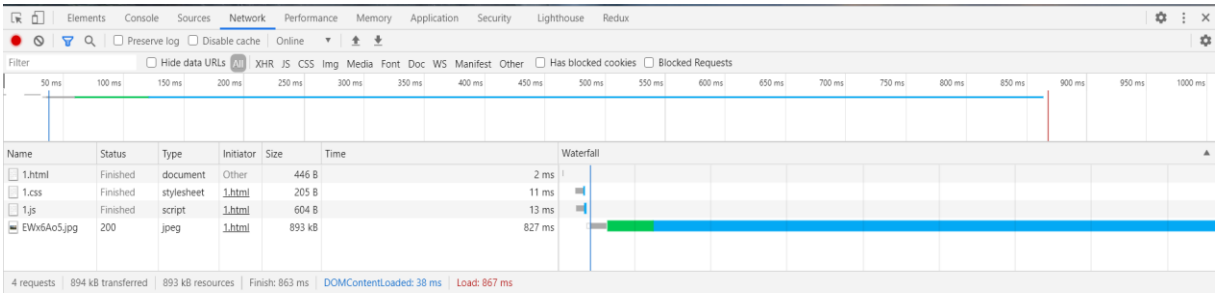

Рисунок 3.2 – Завантаження DOM з 4G

При завантаження сторінки, з швидкістю 3G ситуація вже інша. Час завантаження DOM однаковий, але фонове зображення завантажувалось, 19s. За цей час користувач може покину вашу сторінку, так і не дочекавшись її завантаження.

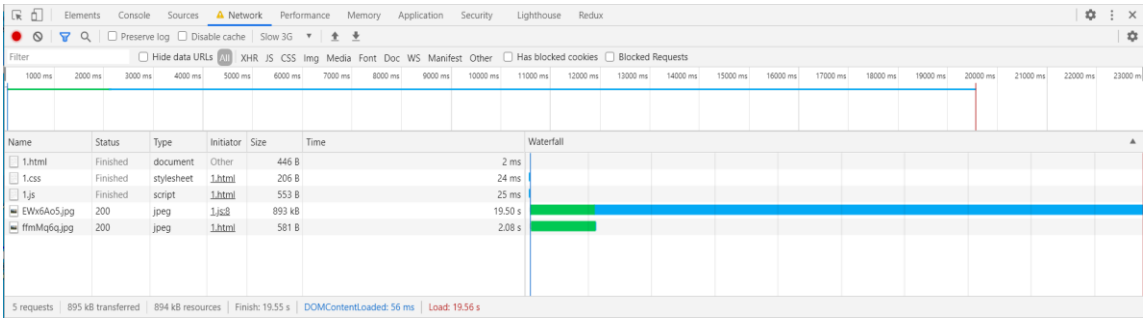

Рисунок 3.3 – Завантаження DOM з 3G.

Щоб вирішити цю проблему, потрібно стиснути фонове зображення. Це простий, та найлегший спосіб збільшити швидкодію даного сайту. Це може збільшити швидкодію завантаження зображення з 19s до 5s.

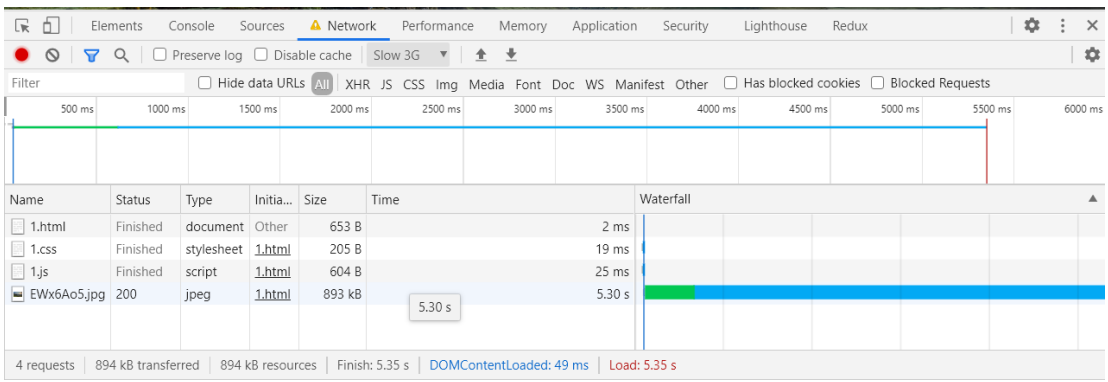

Рисунок 3.4 – Завантаження сайту, із зжатим зображенням, 3G.

Також є і інші методи вирішення даної проблеми. Використати зображення заглушку, замість оригінального. При завантаженні сторінки, використати зображення, розмір якого є 30x30px. З цим, розмір оригінального зображення 7372x4392, зменшиться до 800 байтів.

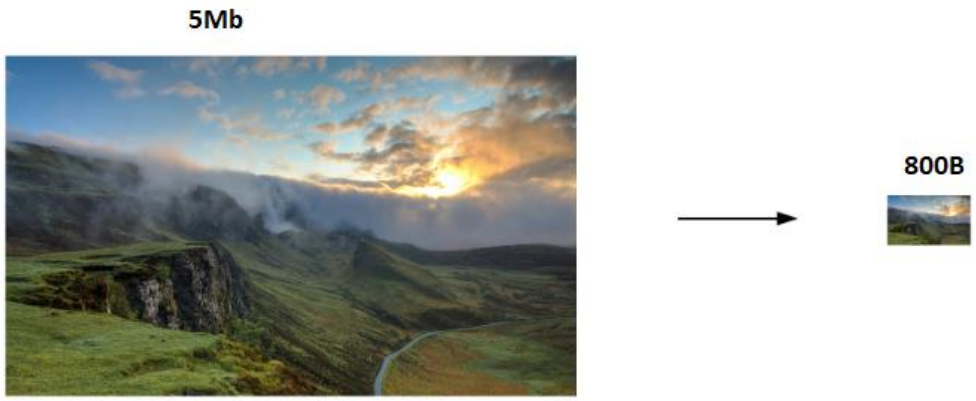

Рисунок 3.5 – Приклад зменшення зображення.

Це дозволить зменшити швидкість загрузки з 5 секунд, до 1.7 мс, як на 4G інтернеті. Після того, як завантажиться наше зображення заглушка, асинхронно на фоні, завантажити оригінальне зображення, та замінити за допомогою JS:

```
img.onload = () => {
 item.classList.remove('asyncImage');
 return item.nodeName === 'IMG' ? item.src = item.dataset.src :
 item.style.backgroundImage = `url(${item.dataset.src})`;
}
```

За допомогою даного JavaScript коду, результати завантаження наступні. Сайт завантажився за 1.72 секунди.

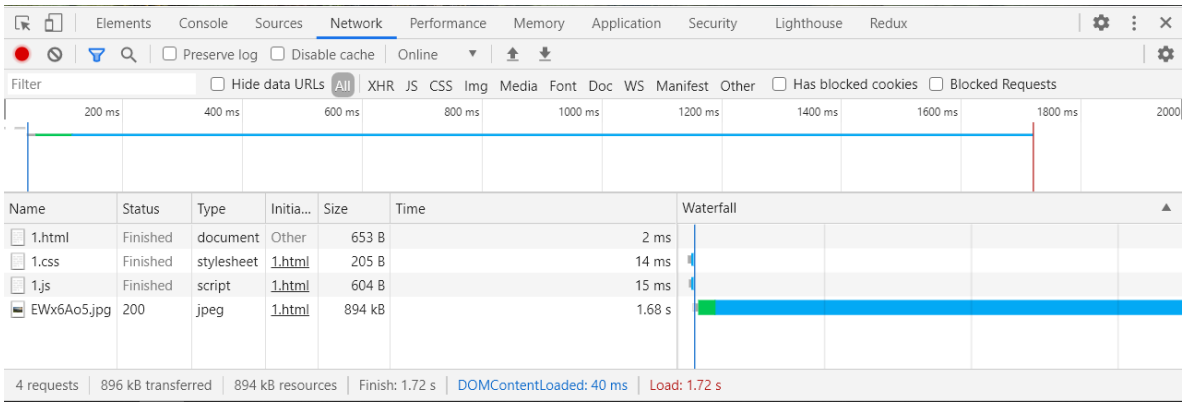

Рисунок 3.6 – Приклад завантаження сайти, після зменшення зображення

В змодельованій ситуації, в веб-додатку, була одна картинка яка важить 5Мб. Але в реальній ситуації, зображень на сайті можуть бути десятки, або навіть сотні. Дому їх оптимізація, це важливий крок до прискорення швидкодії роботи сайту. Це дасть користувачу, комфортне використання вашого сайту, і зменшить випадки завчасного покидання сайту.

3.3. Моделювання методу роботи з SVG зображеннями.

Проведемо дослідження, при перших десяти секундах завантаження сторінки habr.com. На завантаження зображень . в перші 10 секунд, знадобилось 5.1 секунди, це дуже великий результат. Маючи 22 зображення, можна підрахувати, зменшення часу завантаження, при зменшені початкового розміру картинок, для більш швидкої здобуття інтерактивності з користувачем.

Так як SVG зображення це, XML мова розмітки на основі векторної графіки, то вага однієї картинки дуже мала. Візьмемо код готового SVG зображення, та збережемо його у файлі.

Код готового SVG зображення:

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 477.175 477.175">

<path d="M360.731 229.075l-225.1-225.1c-5.3-5.3-13.8-5.3-19.1 0s-5.3 13.8 0 19.1l215.5 215.5-215.5 215.5c-5.3 5.3-5.3 13.8 0 19.1 2.6 2.6 6.1 4 9.5 4 3.4 0 6.9-1.3 9.5-4l225.1-225.1c5.3-5.2 5.3-13.8.1-19z"/>

</svg>

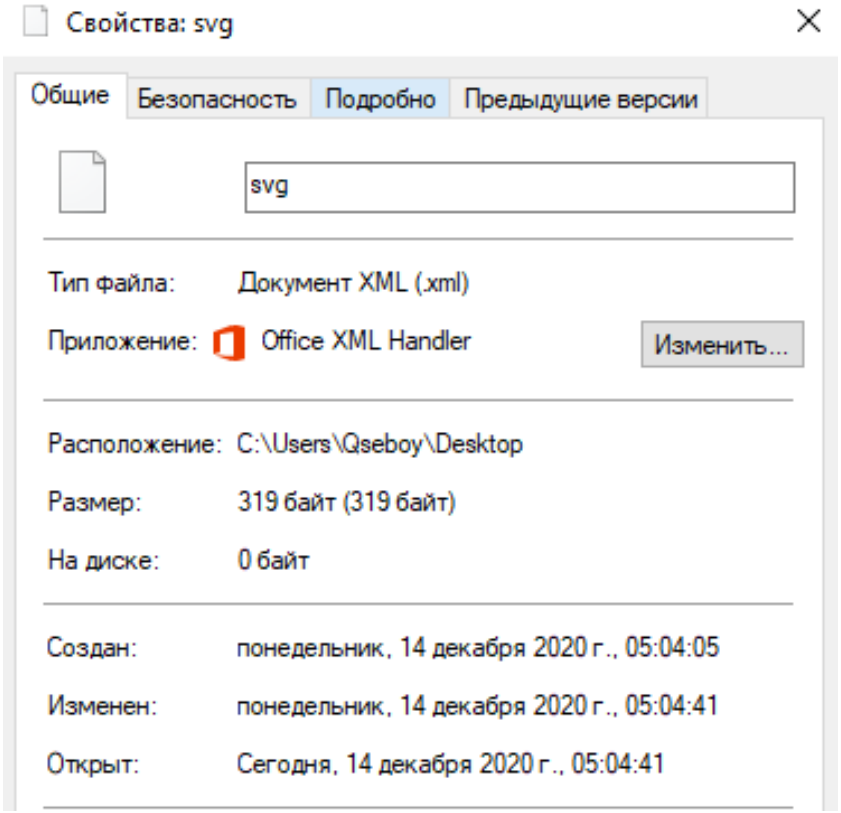

Рисунок 3.7 – Властивості створеного XML документу

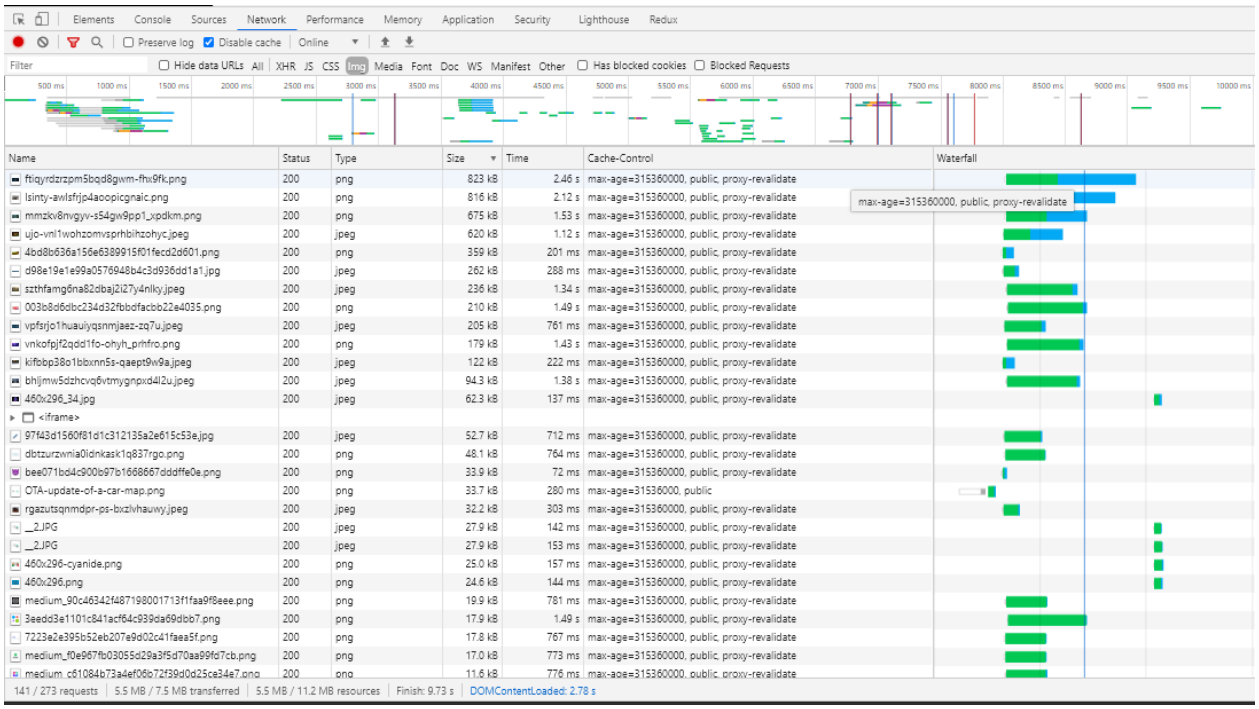

Рисунок 3.8 – Завантаження картинок, ресурсу habr.com за інтервал часу 10 секунд

За даний час, у мережі 4G, передалось по мережі 5.5MB зображень, при цьому на сайт завантажилось 5.5MB.

Приблизний розмір SVG- файлу, 320байт. Для розрахунку, приблизного часу, візьмемо початковий розмір картинки, та їх кількість.

$$S = n * s_1, \quad (3.1)$$

де T – час завантаження всіх картинок;

n- кількість зображень;

s_1 - приблизна вага одного зображення, байт.

$$S = 22 * 320 = 7040 \text{ байт} = 6.865 \text{ КБ} \quad (3.2)$$

Приблизний час завантаження всіх зображень, буде 300 ms. Дана дія зменшить час, завантаження картинок на 4.8s. Це призведе до зменшення часу інтерактивності з користувачем.

Так як швидкість завантаження сайту залежить від типу мережі, який використовує клієнт, та швидкості його інтернету. Тому буде проведений аналогічний тест завантаження сторінки, при використані 3G, 2G типу мобільних мережі. Срощена таблиця швидкості мереж [12].

Таблиця 3.2 – Швидкість типів мобільної мережі.

Технологія	Швидкість в теорії	Швидкість на практиці
2G	474 Кбіт/с	200-300 Кбіт/с
3G	21-42Мбіт/с	10-25 Мбіт/с
4G (LTE)	75Мбіт/с	20-50 Мбіт/с
4G (LTE-A)	150-300Мбіт/С	50-70 Мбіт/с

Проведення аналогічного тесту для мережі 3G. Завантаження ресурсів по мережі, за 10 секунд зіставило 70 KB. При цьому, сайт не став інтерактивним, тому що DOM ще не завантажився.

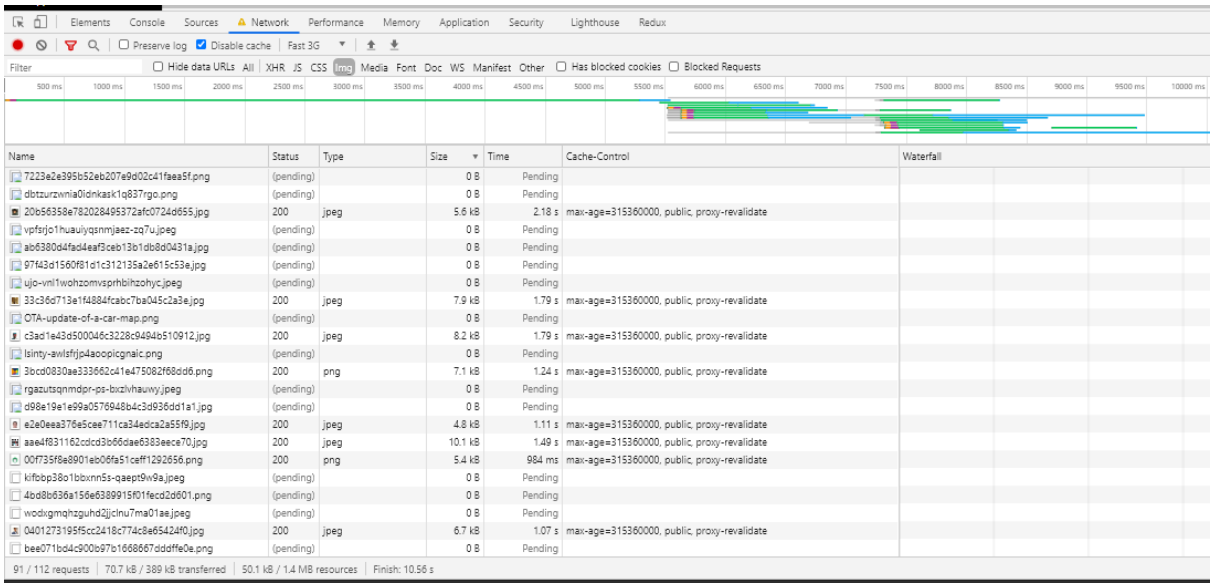

Рисунок 3.9 – Завантаження картинок, ресурсу habr.com за інтервал часу 10 секунд

Проведення аналогічного тесту для мережі 2G. Завантаження ресурсів по мережі 2G, зіставило 629В. За даний час, оригіналі завантажиться 1-2 картинки найменшого розміру.

При використанні методу векторного використання зображення, будуть наступні результати. Розрахуємо швидкість завантаження, за 1 секунду.

$$V = S / t, \tag{3.3}$$

де V – швидкість завантаження ресурсів.

S – загальна вага завантажених файлів;

t – час завантаження, секунди.

$$V = 70{,}7 \text{ КБ} / 10{,}56 = 6.69 = 7 \text{ КБ} \tag{3.4}$$

Знаючі, що вага одного зображення зіставляю приблизно 320 В, розрахуємо кількість зображень, які завантажаться за цей час.

$$N = 7 \text{ КБ} / 320 \text{ В} = 22.4 = 22 \tag{3.5}$$

При використанні даного методу, за час завантаження сайту, можна загрузити 22 картинки, що зменшить час до інтерактивності з користувачем.

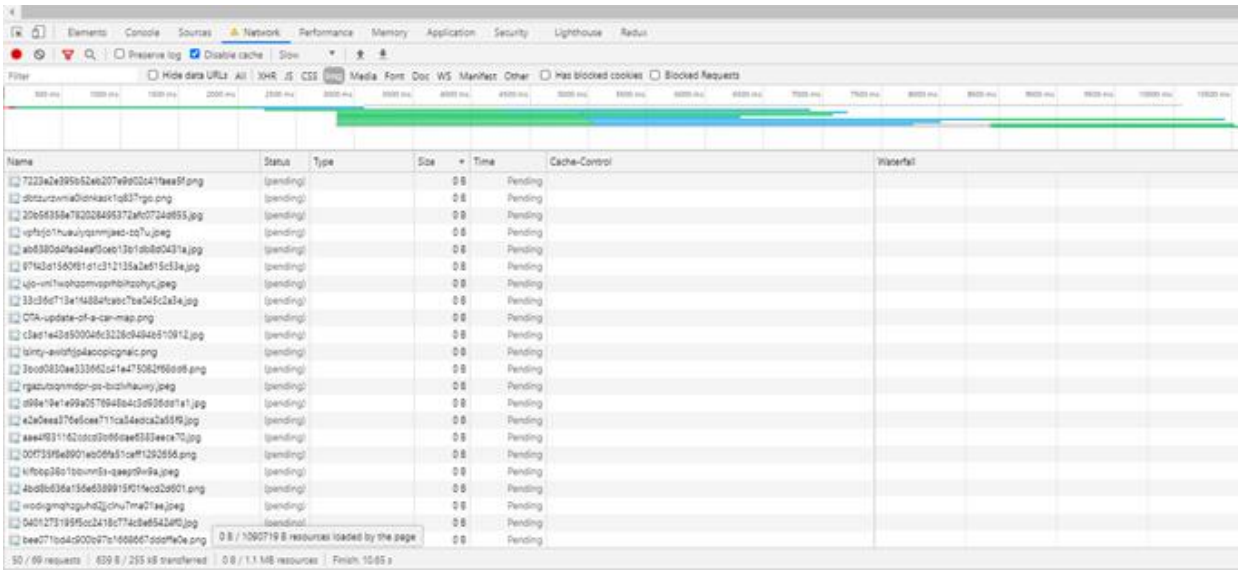

Рисунок 3.10 – Завантаження картинок, ресурсу habr.com за інтервал часу 10 секунд

Для мережі 2G, при використанні методу векторного зображення завантажиться:

$$N = 639B / 320\ B = 1.99 = 2 \quad (3.6)$$

Рисунок 3.8 – Завантаження картинок, ресурсу habr.com за інтервал часу 10 секунд

3.5 Висновки до розділу

У даному розділі були розглянуті методи, своєчасного заміни зображення та метод використання векторного формату, для прискорення завантаження сторінки.

По-перше, можна зробити висновок, що при використані методу своєчасної заміни зображення на більш низької якості я не дуже ефективним. На це вказують отримані данні.

Завантаження сторінки, більш швидше при використанні методу завантаження векторного зображення. Так як векторні зображення, це текст, який браузер перетворює в картинку, тому розмір таких файлів дуже малий, та менший ніж стисканні зображення.

Процес впровадження даного методу нескладний, потрібно проаналізувати свій сайт, знайти ті місця, де зображення пагубна вливають на завантаження, та замінити їх.

Так як тип мереж впливає на швидкість завантаження, то потрібно зробити перше завантаження найбільш меншим по розміру файлів зображення. Тому в даній ситуації, векторний метод є дуже ефективним.

ВИСНОВОК

Отже, у даній роботі був розроблений метод роботи з векторними зображеннями, який дозволяє зменшити загальний час завантаження сторінки, прискорити TTI (time to interactive), що взагалі відобразиться на позитивному опиті користування веб-додатком. Для цього у роботі буди виконані такі дії:

- Проаналізовані більшість програмних методів прискорення веб-додатків. Тому, що програмні методи можна оптимізувати, та вирішити проблемні місця. Що не скажеш про мережевий та апаратні методи прискорення, які потребують багато коштів, для їх оптимізації.

- Проаналізовані існуючи методи, для розрахунку продуктивності, інтерактивності веб-додатків. Важливими даними, було визначення показника відмов, щоб зрозуміти чи є проблема в веб-додатку. За допомогою бюджетів продуктивності можна слідкувати за певними критеріями сайту, і підтримувати їх.

- Проаналізовані існуючи методо роботи з зображеннями, та розгляд їх недоліків та переваг. На основі цих даних, був запропонований метод роботи з векторними зображеннями.

- За допомогою технологій тестування, були проведено тестування, зрівняння часу завантаження сторінки на різних типах мереж, з використанням різних типів роботи з зображеннями, включаючи запропонований векторний метод. Було встановлено зменшення загальної ваги сторінки, за рахунок виростання векторів з 5.5 MB до 6.865 KB.

Таким чином, раздроблений метод буде ефективний на сайтах, де є багато зображень, наприклад сайт з новинами, або онлайн магазини. Недоліком даного методу, є необхідність зберігання копії оригінально зображення, у мережі CND, та на сервері.

СПИСОК ЛІТЕРАТУРИ

1 Програми для стискання зображення, оптимізація загрузки зображення. [Електроні дані]. – Режим доступу https://habr.com/ru/post/482820/. – Заголовок з екрана.

2 React Lazy Load. [Електроні дані]. – Режим доступу https://ru.reactjs.org/docs/code-splitting.html Заголовок з екрана.

3 Build SPA app using the DOM and JavaScript [Електроні дані] - https://itnext.io/build-a-single-page-web-app-javascript-and-the-dom-90c99b08f8a9 Заголовок з екрана.

4 Розгляд фреймворку SvelteJS. [Електроні дані]. – Режим доступу https://habr.com/ru/post/446026//. – Заголовок з екрана.

5 Використання бібліотеки Preact [Електроні дані]. – Режим доступу https://preactjs.com/. – Заголовок з екрана.

6 И.А. Богигін, К.А. Калигін. Дослідження методів збільшення продуктивності WEB-додатків. // С.1-6

7 Що таке CND. [Електроні дані]. – Режим доступу https://habr.com/ru/company/selectel/blog/463915//. – Заголовок з экрана.

8 DNS в картинках. [Електроні дані]. – Режим доступу https://habr.com/ru/post/413515/. Заголовок з екрана.

9 DIG TOOLS. [Електроні дані]. – Режим доступу - https://linux-faq.ru/page/komanda-dig. – Заголовок з екрана.

10 Статистика швидких DNS-серверів [Електроні дані]. – Режим доступу https://www.solvedns.com/dns-comparison/. Заголовок з екрана.

11 Дослідження впливу часу завантаження сайту, на прибуток [Електроні дані]. – Режим доступу https://neilpatel.com/blog/loading-time/ – Заголовок з екрана.

12 Позначення нумерації мобільного інтернету, таблиця швидкостей [Електроні дані]. – Режим доступу https://itblog21.ru/sovety/2086-chto-oznachayut-bukvy-i-tsifry-e-h-4g-v-telefone. – Заголовок з екрана.