Optimizing and Improvement a Web Application Using Open Source Tools

Yuri Kravchenko, Olga Leshchenko, Oleksandr Trush, Nataliia Dakhno and Pavlo Krasnopyorov

Taras Shevchenko National University of Kyiv, Volodymyrs'ka str. 60, Kyiv, 01033, Ukraine

Abstract
This paper analyzes the optimization of a web application using modern open source tools such as Lighthouse and K6. The main goal was to improve the quality and productivity of the additive. The study finally analyzed important metrics such as number of HTTP requests, duration of HTTP requests, HTTP request waits, HTTP requests per second, as well as key indicators such as First Contentful Paint, Largest Contentful Paint, Total Blocking Time, cumulative layout shift and speed index.
The results of this analysis show significant improvements in all the specified metrics, which undoubtedly emphasizes the effectiveness of the optimization methods and tools used. The number of HTTP requests has increased and their duration has decreased, which degrades the overall speed of processing requests. It is important to note that the page load time has become significantly faster, with a significant reduction in First Contentful Paint and Largest Contentful Paint. These improvements not only enhanced the user experience, but also positioned the app as more competitive in the market.

Keywords [1]
WEB APP, optimization, NEXTJS, Lighthouse, K6, YpeScript, JavaScript HTML, CSS

1. Introduction

In today's digital world, web applications remain essential tools for businesses and consumers [1,2]. The speed, performance and user experience of web applications have become key factors in their success. Optimization web applications is an important task for developers to ensure speed of service provision, efficient operation and user satisfaction. The purpose of the work is the research of optimization methods for web application development and the application of effective optimization methods aimed at improving the speed and reactivity of web applications. Web application optimization includes various aspects, from architectural designs to database query optimization and loading page optimization. The results of research and development can be used to improve web applications, ensure fast loading of pages, efficient work with the database and optimal use of resources.

To achieve the goal, the following tasks are solved:

- overview of web application optimization methods;
- web application development;
- evaluation of the effectiveness of the web application;
- an experimental research of web application optimization.

This work is important because the performance and user experience of web applications are critical factors in meeting the needs of today's user. The development of effective methods of optimizing web applications will contribute to the development of the Internet space and the improvement of the quality of web services for users.

2. Overview of web application optimization methods

There are many options for how to optimize a web application, but to group and systematize them, you can use the OSI (Open Systems Interconnection) model [3]. By looking at the OSI model, we can

Information Technology and Implementation (IT&I-2023), November 20-21, 2023, Kyiv, Ukraine
EMAIL: kr34@ukr.net (A.1); olga.leshchenko@knu.ua (A.2); oleksandr.trush@knu.ua (A.3); nataly.dakhno@ukr.net (A.4); pavlokrasnopyorov@knu.ua (A.5)
ORCID: 0000-0002-0281-4396 (A.1); 0000-0002-3997-2785 (A.2); 0000-0002-4188-2850 (A.3); 0000-0003-3892-4543 (A.4)

identify at which network layer optimization techniques can be applied to improve the performance of a web application. Application layer, which is the upper level of the OSI model and plays the most important role in the use of optimization methods. This level includes applications and services that provide users with multi-functional capabilities. Applying optimization techniques at the application layer level has great potential to improve performance, efficiency, and user experience.

Analyzing web application optimization methods involves using a variety of tools to evaluate and analyze the effectiveness of different optimization approaches. This includes both mathematical methods of optimization [4 - 6] and instrumental methods. These tools help you understand which optimization techniques are best to use to improve the performance and speed of your web application.

One of the main analysis tools is application performance analysis. It includes collecting and analyzing server load data, user feedback, page load speed, and other performance metrics [7]. Information obtained from monitoring helps to identify problem areas and potential optimization areas [8, 9]. App performance analysis involves evaluating various metrics that help determine how well the app is performing and how its performance affects the user experience [10, 11].

1. **Response Time**: The average response time can be determined using the mathematical expectation (average) response time for all requests. The equation can look like this:

$$ResponseTime = N\Sigma T, \tag{1}$$

where T - is the response time of each individual request, and N - is the quantity of requests.

2. **Page Load Time**: This metric can be calculated as the sum of the loading time of individual resources (images, CSS, JavaScript, etc.) that are included on the page.

$$\mathrm{PageLoadTime} = \Sigma\mathrm{ResourceLoadTime}, \tag{2}$$

where ResourceLoadTime - is the loading time of each resource.

3. **Resource Usage**: CPU time and memory usage can be measured as the percentage of resources used relative to the maximum available resources.

$$\begin{gathered} CPUUsage = \frac{UsedCPUTime}{TotalCPUTime} * 100\%, \\ MemoryUsage = \frac{UsedMemory}{TotalMemory} * 100\%, \end{gathered} \tag{3}$$

4. **Reliability**: It is possible to use the quantity of errors to determine reliability.

$$Reliability = \frac{NumberofErrors}{TotalInteractions} * 100\%, \tag{4}$$

5. **User Interaction Response Time**: Response time to user interaction can be determined by measuring the time between sending a user request and receiving a response.

$$UserInteractionResponseTime = EndTime - StartTime, \tag{5}$$

where EndTime - is the time of receiving a response, and StartTime - is the time of sending a request.

These equations represent general approaches to measuring various performance metrics. Specific equations and measurement metrics can be adapted depending on the application and measurement methodology.

2.1. Tools for web application performance analysis

Paragraph text. Paragraph text. Analysis tools play an important role in the process of optimizing web applications. They provide us with the opportunity to get detailed information about the performance and efficiency of our application, identify problem areas and find ways to solve them.

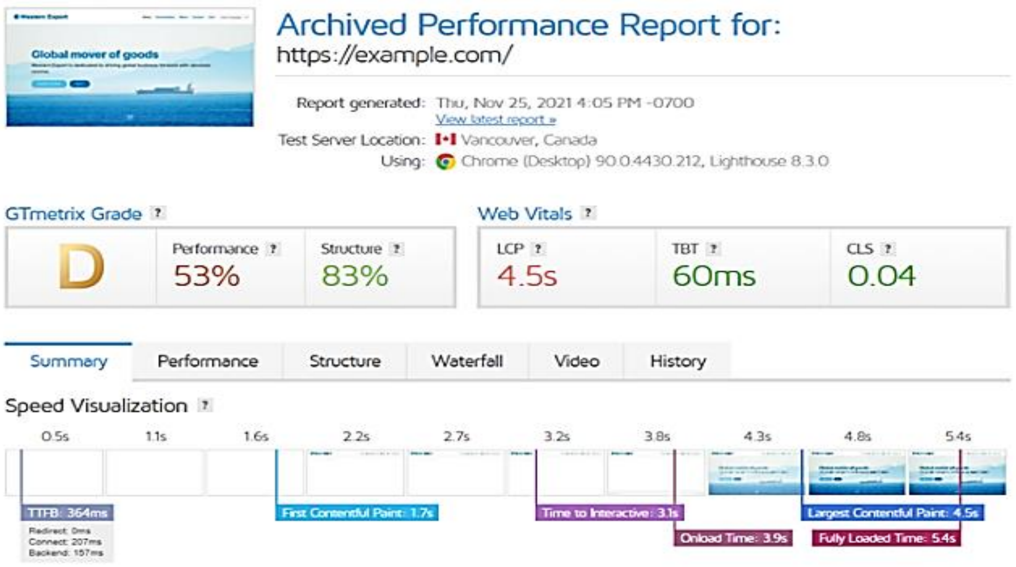
First of all, analysis tools allow you to monitor the performance of a web application in real time [12]. They provide collection and visualization of key metrics such as server recall time, page load time, memory usage, and more. This allows you to identify speed issues that may affect the user experience.

Next, analysis tools provide the ability to perform detailed audits of web applications in terms of loading speed, resource size, caching, and other factors. They help identify problem areas that can be optimized, such as reducing file size, using caching to reduce server requests, etc.

In addition, the analysis tools provide the ability to conduct load tests that allow you to simulate heavy loads on a web application and evaluate its performance and stability. This allows you to identify problems of scalability, insufficient optimization or instability of the system.

Popular web application analysis and optimization tools include:

1. **k6** - is a high-performance tool for load testing and performance verification [13]. It allows developers and engineers to test the scaling of web applications and network services and evaluate their performance under different loads. Benefits: An easy-to-use tool for load testing and performance verification. Supports JavaScript scripting and provides detailed reports. Disadvantages: Some advanced features may only be available in the commercial version. Figure 1 shows an example of use.
2. **GTmetrix:** It is an online tool that provides a detailed report on the performance of web pages. GTmetrix evaluates page load speed, file size, quantity of server requests and other metrics. It also provides optimization recommendations to improve performance [14]. Benefits: Provides in-depth performance analysis, including download speed estimates, image optimization, caching, and other tips. It has a user-friendly interface and supports many test locations. Disadvantages: Some features are only available in the paid version. Reports can be a bit complicated for beginners. Figure 2 shows an example of use.



Archived Performance Report for:
https://example.com/

Report generated: Thu, Nov 25, 2021 4:05 PM -0700
View latest report »
Test Server Location: Vancouver, Canada
Using: Chrome (Desktop) 90.0.4430.212, Lighthouse 8.3.0

GTmetrix Grade: D
Performance: 53%
Structure: 83%

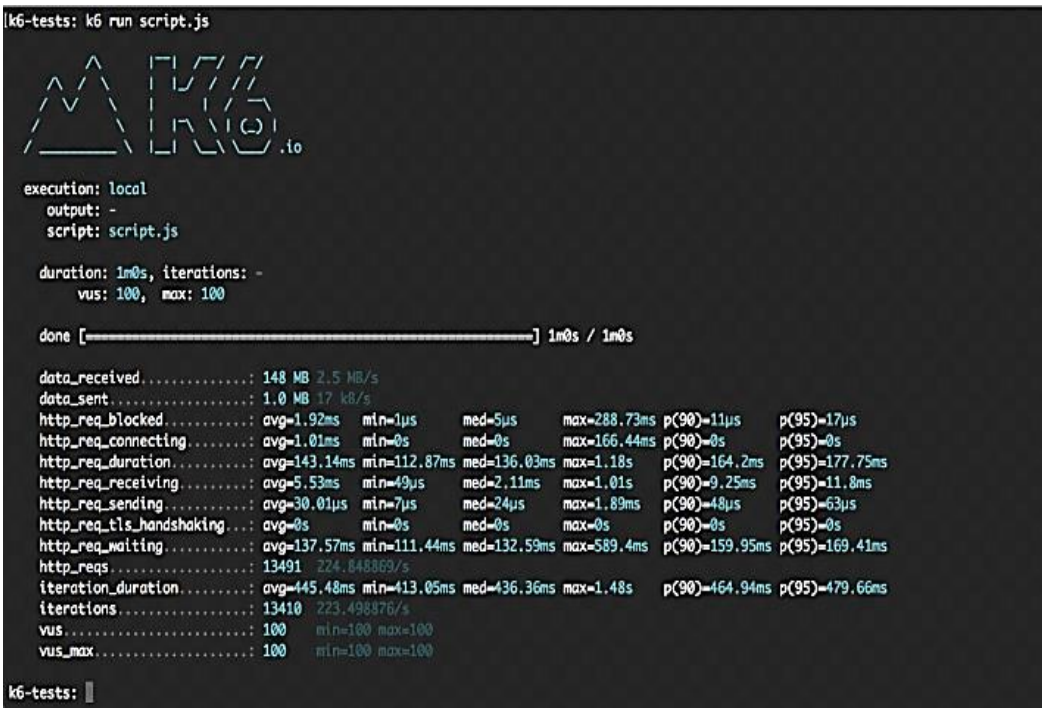
Web Vitals
LCP: 4.5s
TBT: 60ms
CLS: 0.04

Figure 1: An example of the use of k6

1. **WebPageTest:** It is a tool that allows you to test the loading speed of web pages from different locations around the world. It provides detailed information on load hours, page size, server requests, and other metrics [15]. WebPageTest also allows you to run repeated tests to monitor hourly performance. Benefits: Provides detailed performance reports including load speed, page load analysis, query waterfall and other metrics. Allows you to choose the test location and different configurations. Disadvantages: The interface may seem difficult for beginners. There are a limited quantity of free requests. Figure 3 shows an example of use.
2. **Pingdom:** It is a performance monitoring tool that provides information on page load hours, file sizes, quantity of requests, and other metrics [16]. He has too the ability to monitor site performance on an ongoing basis and send notifications about any problems. Pros
3. An easy-to-use tool that provides information about website loading speed and availability. It has a user-friendly interface and supports many test locations. Disadvantages: Limited functionality compared to other tools. Some advanced features are only available in the paid version. Figure 4 shows an example of use.
4. **YSlow**: This is a browser extension that provides web page performance scores based on Yahoo's recommendations [17]. YSlow analyzes various aspects, including caching, file compression, CDN usage, and more, and provides recommendations for improving performance. Benefits: Analyzes web page performance, offers optimization tips such as resource compression, caching, and other

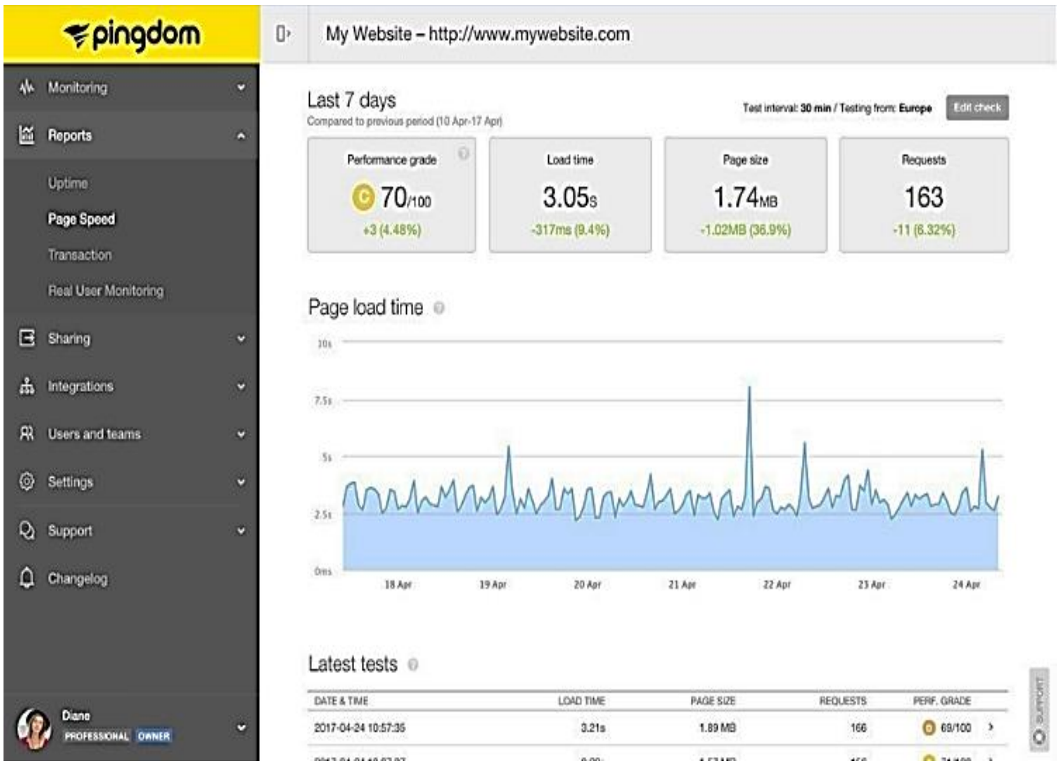
improvements. Integrated with Firebug browser extension. Disadvantages: The development of the tool has been frozen, so it may be less relevant compared to newer tools. Figure 5 shows an example of use.

5. **Apache JMeter:** It is a tool for testing server performance and load. It allows simulation of a large volume of requests to a web application, which helps to evaluate its performance and identify problem areas [18]. Pros: Powerful performance and load testing tool. Supports many protocols and can simulate different load scenarios. Disadvantages: Requires study and experience to use effectively. Does not provide a visual report, requires analysis of results. Figure 6 shows an example of use.



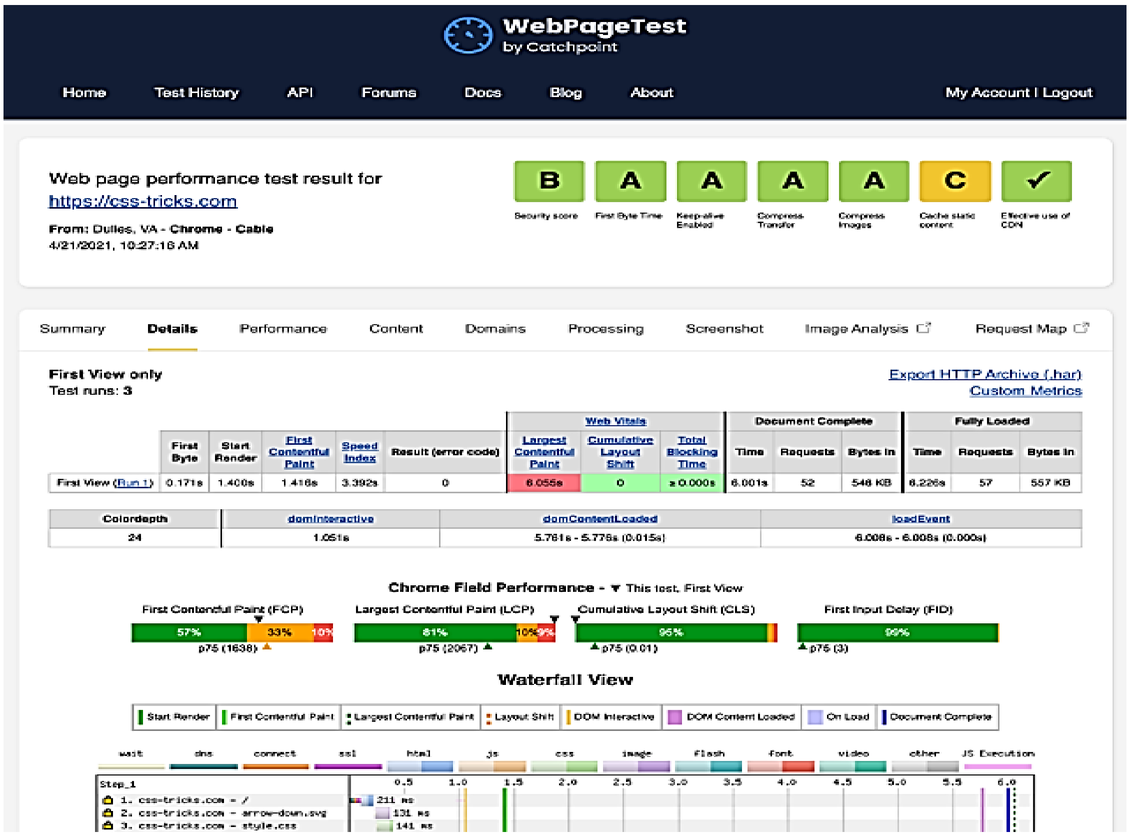
```
k6-tests: k6 run script.js

  execution: local
     output: -
     script: script.js

    duration: 1m0s, iterations: -
         vus: 100,  max: 100

    done [==========================================================] 1m0s / 1m0s

    data_received..............: 148 MB 2.5 MB/s
    data_sent..................: 1.0 MB 17 kB/s
    http_req_blocked...........: avg=1.92ms   min=1µs      med=5µs      max=288.73ms p(90)=11µs     p(95)=17µs
    http_req_connecting........: avg=1.01ms   min=0s       med=0s       max=166.44ms p(90)=0s       p(95)=0s
    http_req_duration..........: avg=143.14ms min=112.87ms med=136.03ms max=1.18s    p(90)=164.2ms  p(95)=177.75ms
    http_req_receiving.........: avg=5.53ms   min=49µs     med=2.11ms   max=1.01s    p(90)=9.25ms   p(95)=11.8ms
    http_req_sending...........: avg=30.01µs  min=7µs      med=24µs     max=1.89ms   p(90)=48µs     p(95)=63µs
    http_req_tls_handshaking...: avg=0s       min=0s       med=0s       max=0s       p(90)=0s       p(95)=0s
    http_req_waiting...........: avg=137.57ms min=111.44ms med=132.59ms max=589.4ms  p(90)=159.95ms p(95)=169.41ms
    http_reqs..................: 13491  224.848869/s
    iteration_duration.........: avg=445.48ms min=413.05ms med=436.36ms max=1.48s    p(90)=464.94ms p(95)=479.66ms
    iterations.................: 13410  223.498876/s
    vus........................: 100    min=100 max=100
    vus_max....................: 100    min=100 max=100

k6-tests:
```

Figure 2: Example of using GTmetrix



DATE & TIME	LOAD TIME	PAGE SIZE	REQUESTS	PERF. GRADE
2017-04-24 10:57:35	3.21s	1.89 MB	166	69/100

Figure 3: Using WebPageTest



Web page performance test result for
https://css-tricks.com
From: Dulles, VA - Chrome - Cable
4/21/2021, 10:27:16 AM

B	A	A	A	A	C	✓
Security score	First Byte Time	Keep-alive Enabled	Compress Transfer	Compress Images	Cache static content	Effective use of CDN

First View only
Test runs: 3

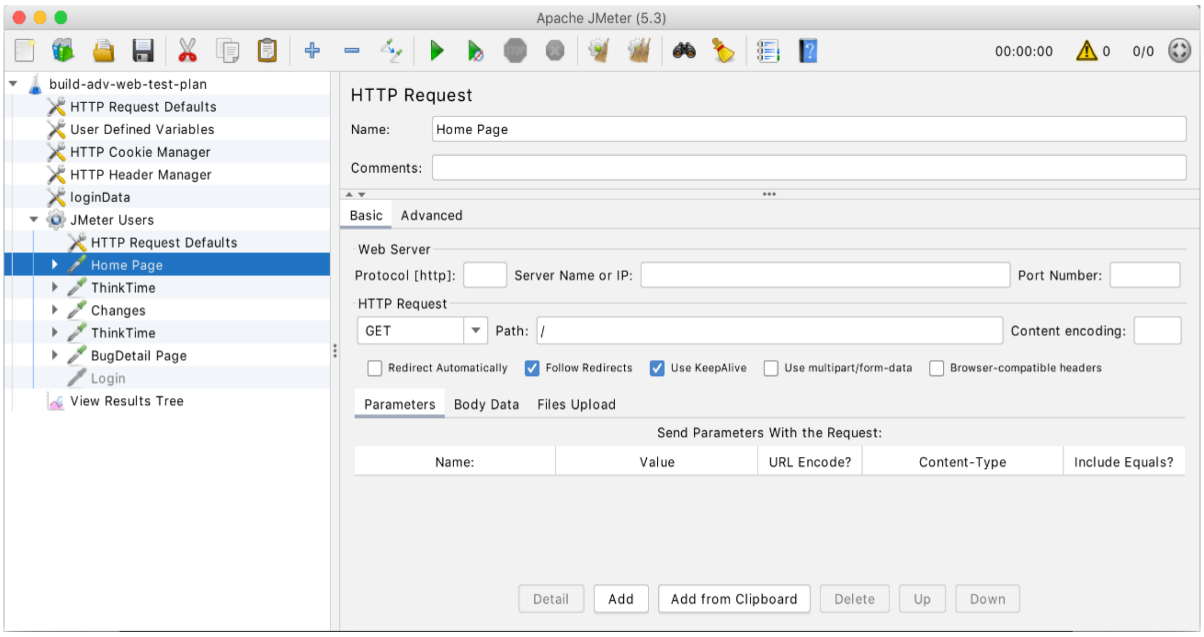
Export HTTP Archive (.har)
Custom Metrics

						Web Vitals			Document Complete			Fully Loaded		
	First Byte	Start Render	First Contentful Paint	Speed Index	Result (error code)	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time	Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 1)	0.171s	1.400s	1.416s	3.392s	0	6.055s	0	≥ 0.000s	6.001s	52	546 KB	6.226s	57	557 KB

Colordepth	domInteractive	domContentLoaded	loadEvent
24	1.051s	5.761s - 5.776s (0.015s)	6.008s - 6.008s (0.000s)

Figure 4: Pingdom usage

These tools help you analyze and optimize the performance of your web applications to provide a better user experience and faster page load times. Thanks to the analysis tools, there is confidence that the web application is running at optimal performance and provides a fast and convenient user experience. Considering these factors, Lighthouse [18, 19] and k6 were chosen as effective tools for query testing in a commercial web application. These tools provide ease of use, flexibility in test setup, analysis of results, and extensibility. Help identify problems, analyze test results, and make informed decisions about app optimization.



HTTP Request

Name: Home Page

Comments:

Basic | Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

GET Path: / Content encoding:

☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?

Figure 5: An example of using Apache JMeter

3. Development of a web application

Web application development includes two main components: client-side and server-side development. Each of these components has its own unique requirements and tasks that are necessary for the successful implementation of the project.

The client part of the web application corresponds to the interaction with the user and the display of information on his devices. Technologies such as HTML, CSS and JavaScript programming language, as well as modern frameworks and libraries that simplify the work of creating an interactive interface are used to develop the client part.

The server part of the web application is responsible for processing user requests, saving and retrieving data from databases, as well as for the application's business process logic.

When developing an optimized web application, the Next.js framework will be used, which is one of the popular and powerful frameworks for developing React-based web applications. One of the main advantages of Next.js is that it works both on the client side and on the server side, which makes it an ideal choice for optimizing applications.

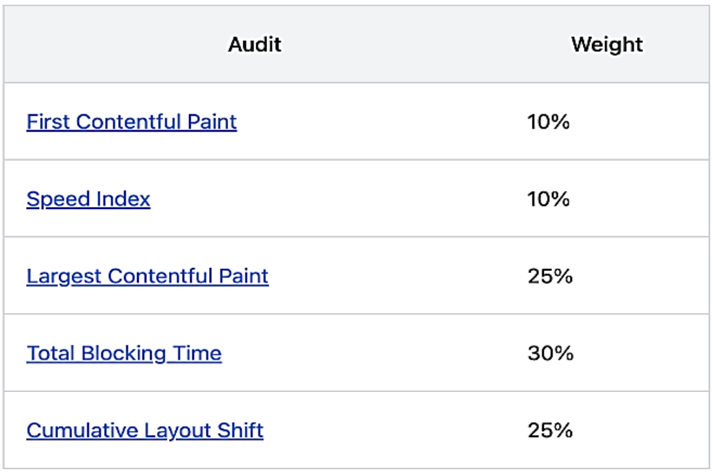
A commercial web application was chosen for the optimization study. Here are some reasons that explain the choice: increased conversions, large volumes of data, SEO and search ranking [20].

Choosing a commercial web application allows you to uncover the most possible optimization methods, after which it requires attention to speed, performance, security, user experience and other factors that are crucial.

4. Evaluation of the effectiveness of the web application

Web application development includes two main components: client-side and server-side development. Each of these components has its own unique requirements and tasks that are necessary for the successful implementation of the project.

To evaluate the performance of the web application, we will use Lighthouse, which is an open tool for analyzing the performance and quality of web applications, as well as K6 to create a stress test. Lighthouse was developed by the Chrome DevTools team and provides tools for evaluating performance, accessibility, SEO optimization, and other aspects of a web application [21, 22]. The main function of Lighthouse is to automatically audit a web page using a set of rules and recommendations. The Lighthouse performance score is a weighted average of metrics, with more weighted metrics having a greater impact on the overall performance score. Scores of indicators are not displayed in the report, but are calculated according to the formula shown in Figure 6.



Audit	Weight
First Contentful Paint	10%
Speed Index	10%
Largest Contentful Paint	25%
Total Blocking Time	30%
Cumulative Layout Shift	25%

Figure 6: Indicator and its weight in efficiency calculation

Key performance metrics that can be measured include:

1. **First Contentful Paint (FCP):** This is the time it takes to display the first content on the page. It can be text, images or other elements. The smaller the FCP value, the faster the web page will display the first content. Figure 7 shows how the metric value is interpreted for loading.
The FCP is calculated as the difference between the FCP time and the initial page load time:

$$FCP = \mathrm{tFCP} - tstart, \tag{6}$$

2. **Cumulative Layout Shift (CLS):** This is a metric that measures how unstable the elements on the page are when loading. It measures how much the page layout changes on load, which can lead to an unpleasant user experience, such as when they try to click on an element but it suddenly shifts. CLS is calculated as the sum of the cumulative displacements of objects on the page during loading. Each change in the position of the object is taken into account with a weighting factor depending on the visibility of the object and the size of the page. Mathematically, this can be expressed as:

$$CLS = \Sigma\,(impact\ fraction * distance\ fraction * distance), \tag{7}$$

where:

- Impact fraction - fraction of displacement caused by a change in the object's position.
- Distance fraction - the fraction of the visible area of the page that was affected by the change in the object's position.
- Distance - the actual distance the object moved.

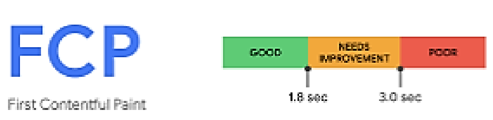


Figure 7: Metric FCP

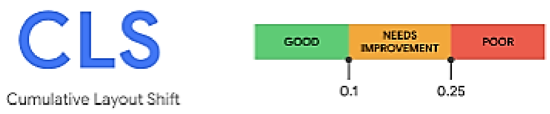


Figure 8: Metric CLS

3. **Largest Contentful Paint (LCP):** This is the time it takes to display the largest piece of content on the page. It can be a large image, video or other important element that attracts the user's attention. Figure 8 shows how the metric value is interpreted for loading.
The LCP is calculated as the difference between the LCP time and the initial page load time:

$$LCP = tLCP - tstart, \tag{8}$$

- t_start - the time the page started to load,
- t_LCP - the time when the largest content becomes visible.

4. **Total Blocking Time (TBT)**: This is the time when the page is blocked and unavailable for user interaction due to the execution of JavaScript code. Figure 9 shows how the metric value is interpreted for loading.
The mathematical model for TBT can be expressed as follows:

- Blocking Time for each task (such as JavaScript or rendering): The amount of time the page is blocked during a specific task. Let's denote this time as BT_i, where i is the index of a separate task.
- Quantity of tasks (N): Quantity of all tasks that blocked the page.
- Total Blocking Time, TBT: This is the total block time for all tasks:

$$TBT = \Sigma BTi, \text{де } i \text{ від } 1 \text{ до } N, \tag{9}$$

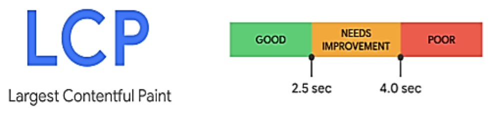


Figure 9: Metric LCP

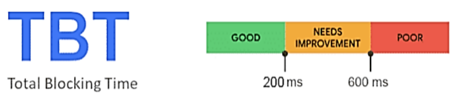
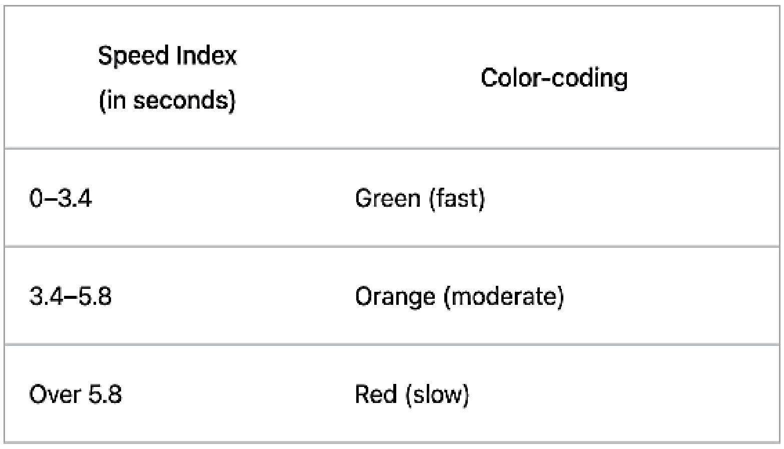


Figure 10: Metric TBT

5. **Speed Index:** This is the time that shows how quickly the content is displayed visually when the page loads. First, Lighthouse captures a video of the page loading in the browser and calculates the visual transition between frames, then uses the Speedline Node.js module to calculate the speed index. Figure 11 shows how the metric value is interpreted for loading.

Taking into account these optimization metrics helps to make an objective assessment of the performance of the web application. Comparing these metrics to set goals and standards will help identify issues and make optimizations to improve performance and user experience.



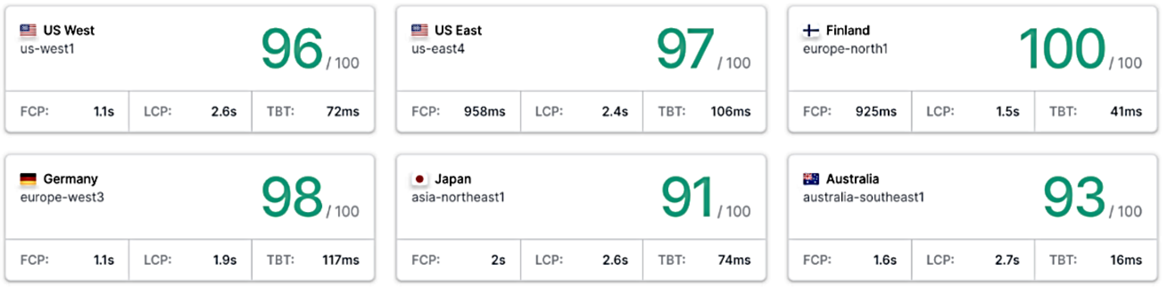
Speed Index (in seconds)	Color-coding
0–3.4	Green (fast)
3.4–5.8	Orange (moderate)
Over 5.8	Red (slow)

Figure 11: Interpretation of the Speed Index by time

5. Experimental study of optimization methods using Lighthouse and K6 tools

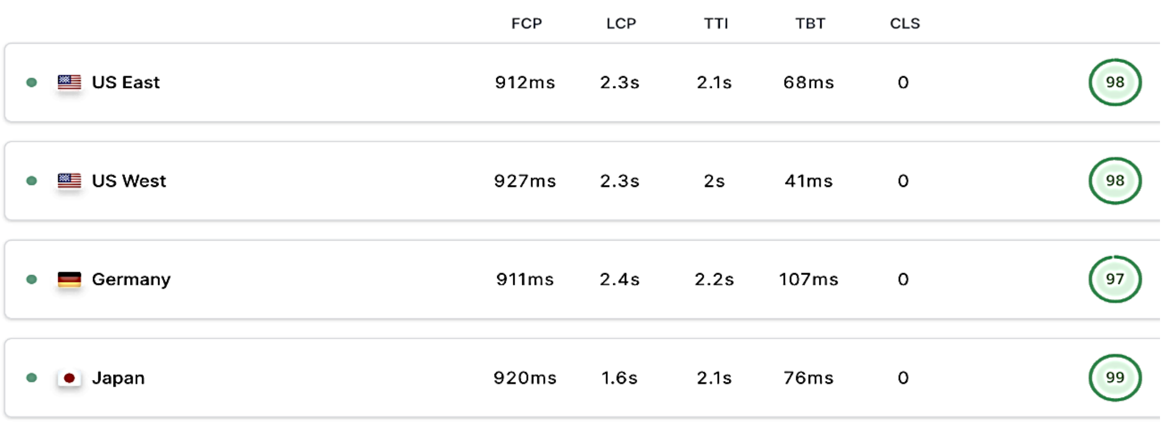
An experimental study of the optimization of a web application can include several stages, and the main emphasis will be on using the Lighthouse tool for local testing [23] and conducting tests in different regions for hosting on the domain where the web application is located, as well as stress testing using tools K6 [24, 25]. The results of testing in different regions on the main page are in Figure 12 and Figure 13, respectively.

Performance testing for different regions is an important part of the web application optimization process. It helps ensure global availability, improve user experience and determine the optimal hosting settings for the application.



Region	Score	FCP:	LCP:	TBT:
US West us-west1	96 / 100	1.1s	2.6s	72ms
US East us-east4	97 / 100	958ms	2.4s	106ms
Finland europe-north1	100 / 100	925ms	1.5s	41ms
Germany europe-west3	98 / 100	1.1s	1.9s	117ms
Japan asia-northeast1	91 / 100	2s	2.6s	74ms
Australia australia-southeast1	93 / 100	1.6s	2.7s	16ms

Figure 12: Lighthouse test results in different regions of the main page [7]

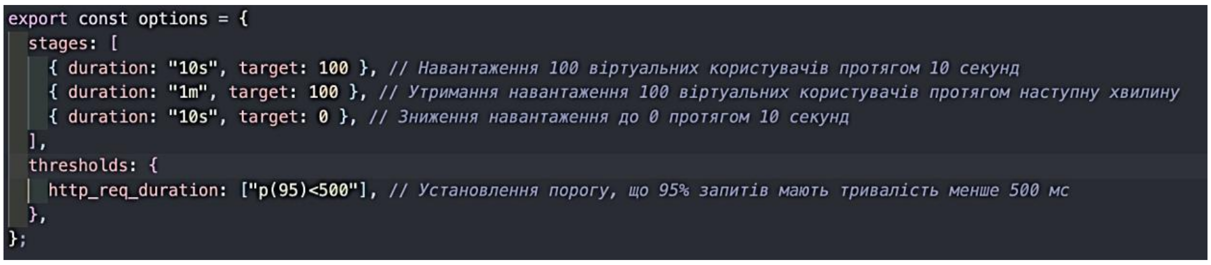


	FCP	LCP	TTI	TBT	CLS	
US East	912ms	2.3s	2.1s	68ms	0	98
US West	927ms	2.3s	2s	41ms	0	98
Germany	911ms	2.4s	2.2s	107ms	0	97
Japan	920ms	1.6s	2.1s	76ms	0	99

Figure 13: Lighthouse test results in different regions of the main page [7]

The K6 tool was chosen for the stress test - it is a very powerful tool for stress testing and loading web applications and APIs. It is designed to help developers and engineers test the speed and stability of their systems under heavy load.

One of the advantages of the K6 is its ease of use. It has a simple syntax that allows you to quickly create and configure tests. K6 is written in the Go programming language, which makes it fast and efficient. It also has built-in support for JavaScript, allowing you to use your own code to create complex test scenarios. The configuration of the script with comments is presented in Figure 14.



```
export const options = {
  stages: [
    { duration: "10s", target: 100 }, // Навантаження 100 віртуальних користувачів протягом 10 секунд
    { duration: "1m", target: 100 }, // Утримання навантаження 100 віртуальних користувачів протягом наступну хвилину
    { duration: "10s", target: 0 }, // Зниження навантаження до 0 протягом 10 секунд
  ],
  thresholds: {
    http_req_duration: ["p(95)<500"], // Установлення порогу, що 95% запитів мають тривалість менше 500 мс
  },
};
```

Figure 14: Script configuration for testing requests

Uses the http module from K6 to make a GET request to a web application URL.

The Options object contains the K6 configuration parameters. We define different stages (stages) for the load, increasing it to 100 virtual users for 10 seconds, maintaining it at this level for the next minute, and then reducing it to 0 for another 10 seconds. The Thresholds object allows you to set threshold values for metrics. We use http_req_duration (duration of requests) and set the threshold that 95% of requests have a duration of less than 500 ms. The results of the k6 web application before and after the application of optimization methods are shown in Figures 16 and 17, respectively.

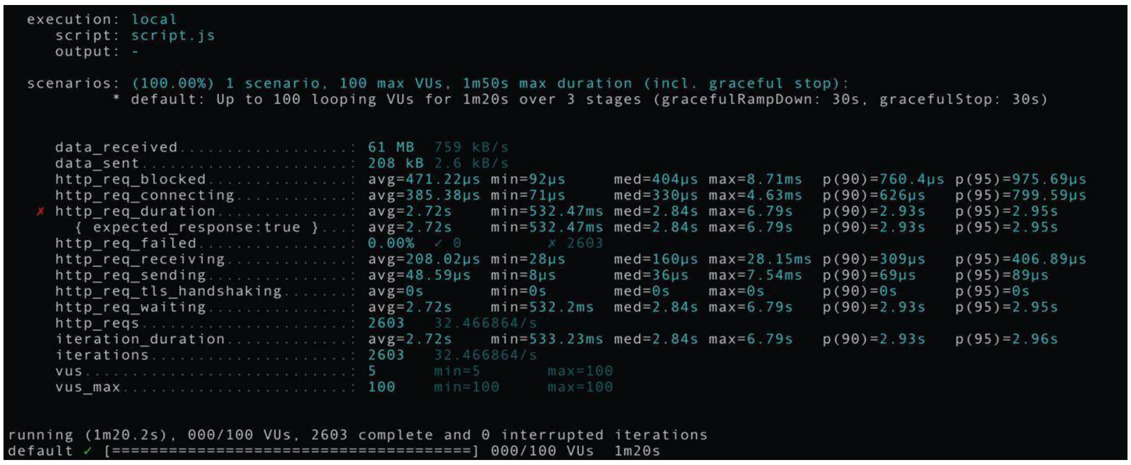
When using the K6 tool, the following results were obtained:

1. Quantity of HTTP requests. Includes all successful and failed requests. 2603 queries were executed before optimization, and 20222 queries were received after using the K6 tool. The increase indicates that the optimization of the application contributed to more efficient processing of requests and resources, which helps to reduce the load on the server.

2. HTTP request duration. Before optimization, it was 2.72 seconds, and after using the K6 tool, 312.17ms was obtained. The duration of requests decreased by 88.43% after optimization. The reduction indicates that the app has become more responsive and responds quickly to user requests, which improves the overall user experience.

3. Waiting for HTTP requests. Before optimization, it was 2.72 s, and after using the K6 tool, 312.92 ms was obtained. Expectation decreased by 88.29%. The reduction indicates optimization of the application's network interaction, which contributes to higher performance and responsiveness.

4. Quantity of HTTP requests per second. Before optimization, it was 32.46 requests/s, and after using the K6 tool, 252.76 requests/s were obtained. The 678.6% increase shows the app's ability to serve more users simultaneously and improves its scalability.



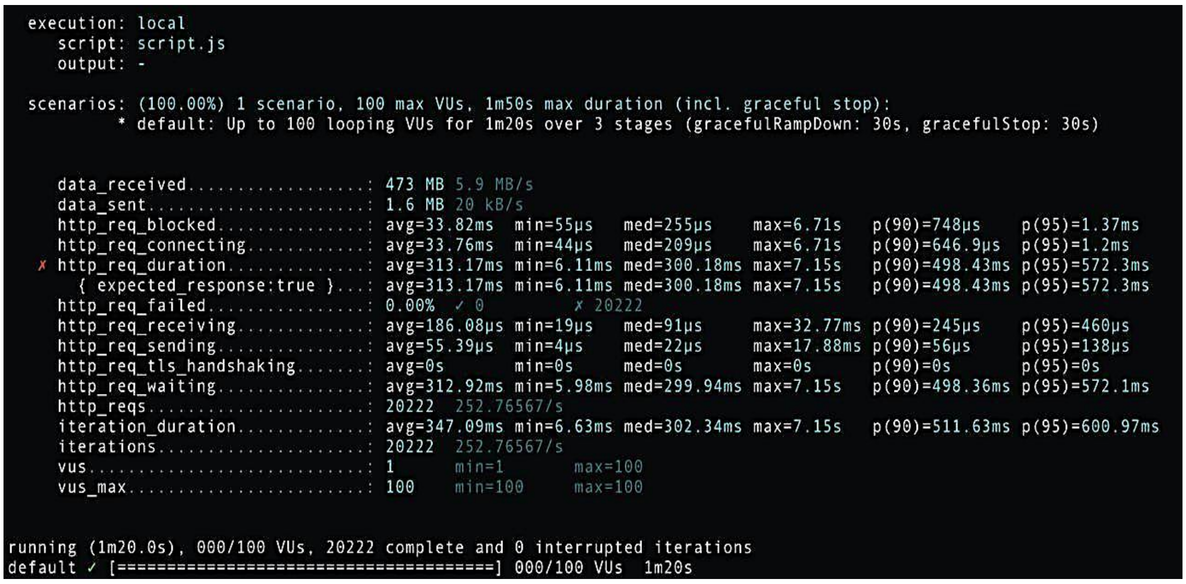
```
  execution: local
     script: script.js
     output: -

  scenarios: (100.00%) 1 scenario, 100 max VUs, 1m50s max duration (incl. graceful stop):
           * default: Up to 100 looping VUs for 1m20s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)


     data_received..................: 61 MB  759 kB/s
     data_sent......................: 208 kB 2.6 kB/s
     http_req_blocked...............: avg=471.22µs min=92µs     med=404µs max=8.71ms  p(90)=760.4µs p(95)=975.69µs
     http_req_connecting............: avg=385.38µs min=71µs     med=330µs max=4.63ms  p(90)=626µs   p(95)=799.59µs
   ✗ http_req_duration..............: avg=2.72s    min=532.47ms med=2.84s max=6.79s   p(90)=2.93s   p(95)=2.95s
       { expected_response:true }...: avg=2.72s    min=532.47ms med=2.84s max=6.79s   p(90)=2.93s   p(95)=2.95s
     http_req_failed................: 0.00%  ✓ 0         ✗ 2603
     http_req_receiving.............: avg=208.02µs min=28µs     med=160µs max=28.15ms p(90)=309µs   p(95)=406.89µs
     http_req_sending...............: avg=48.59µs  min=8µs      med=36µs  max=7.54ms  p(90)=69µs    p(95)=89µs
     http_req_tls_handshaking.......: avg=0s       min=0s       med=0s    max=0s      p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=2.72s    min=532.2ms  med=2.84s max=6.79s   p(90)=2.93s   p(95)=2.95s
     http_reqs......................: 2603   32.466864/s
     iteration_duration.............: avg=2.72s    min=533.23ms med=2.84s max=6.79s   p(90)=2.93s   p(95)=2.96s
     iterations.....................: 2603   32.466864/s
     vus............................: 5      min=5       max=100
     vus_max........................: 100    min=100     max=100


running (1m20.2s), 000/100 VUs, 2603 complete and 0 interrupted iterations
default ✓ [======================================] 000/100 VUs  1m20s
```

Figure 15: Output of K6 tool information to main page queries before optimization

After carrying out two stress tests for the web application - before optimization and after optimization, we can draw the following conclusions: performance has improved, loading time has decreased, resource consumption has decreased, and scalability has improved.

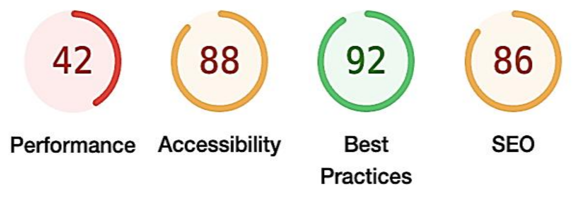
At the next stage, the Image component and Content Delivery Network (CDN) were used to improve the speed and performance of web applications. Figure 17 shows the Lighthouse results for a web application without adding optimization methods, and Figure 18 shows the values of these metrics.

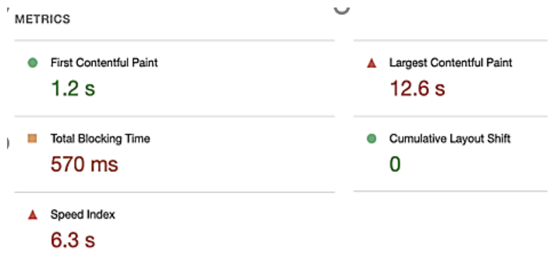
Figure 19 shows the Lighthouse results for the web application after adding optimization methods, and Figure 21 shows the values of these metrics



```
  execution: local
     script: script.js
     output: -

  scenarios: (100.00%) 1 scenario, 100 max VUs, 1m50s max duration (incl. graceful stop):
           * default: Up to 100 looping VUs for 1m20s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)


     data_received..................: 473 MB 5.9 MB/s
     data_sent......................: 1.6 MB 20 kB/s
     http_req_blocked...............: avg=33.82ms  min=55µs    med=255µs    max=6.71s    p(90)=748µs    p(95)=1.37ms
     http_req_connecting............: avg=33.76ms  min=44µs    med=209µs    max=6.71s    p(90)=646.9µs  p(95)=1.2ms
   ✗ http_req_duration..............: avg=313.17ms min=6.11ms med=300.18ms max=7.15s    p(90)=498.43ms p(95)=572.3ms
       { expected_response:true }...: avg=313.17ms min=6.11ms med=300.18ms max=7.15s    p(90)=498.43ms p(95)=572.3ms
     http_req_failed................: 0.00%  ✓ 0           ✗ 20222
     http_req_receiving.............: avg=186.08µs min=19µs    med=91µs     max=32.77ms p(90)=245µs    p(95)=460µs
     http_req_sending...............: avg=55.39µs  min=4µs     med=22µs     max=17.88ms p(90)=56µs     p(95)=138µs
     http_req_tls_handshaking.......: avg=0s       min=0s      med=0s       max=0s      p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=312.92ms min=5.98ms med=299.94ms max=7.15s    p(90)=498.36ms p(95)=572.1ms
     http_reqs......................: 20222  252.76567/s
     iteration_duration.............: avg=347.09ms min=6.63ms med=302.34ms max=7.15s    p(90)=511.63ms p(95)=600.97ms
     iterations.....................: 20222  252.76567/s
     vus............................: 1      min=1       max=100
     vus_max........................: 100    min=100     max=100


running (1m20.0s), 000/100 VUs, 20222 complete and 0 interrupted iterations
default ✓ [======================================] 000/100 VUs  1m20s
```

Figure 16: Output of K6 tool information to main page queries after optimization





METRICS

First Contentful Paint
1.2 s

Largest Contentful Paint
12.6 s

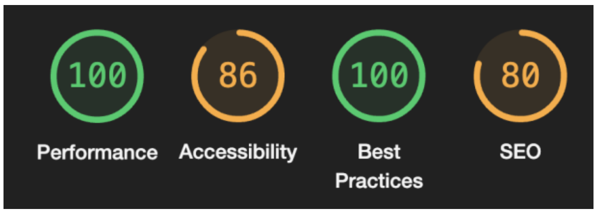
Total Blocking Time
570 ms

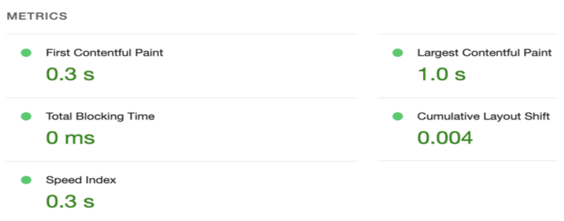
Cumulative Layout Shift
0

Speed Index
6.3 s

Figure 17: Results of the basic web application

Figure 18: Web application metrics values





METRICS

First Contentful Paint
0.3 s

Largest Contentful Paint
1.0 s

Total Blocking Time
0 ms

Cumulative Layout Shift
0.004

Speed Index
0.3 s

Figure19: Results of the web application after adding optimization elements

Figure 20: Values of web application metrics after adding optimization methods

Running the Lighthouse tool produced the following results:

- First Contentful Paint (FCP) reduced by 75%, from 1.2 seconds to 0.3 seconds.
- Largest Contentful Paint (LCP) decreased by 92%, from 12.6 seconds to 1.0 seconds.
- Total Blocking Time (TBT) reduced from 570 milliseconds to 0 milliseconds.

- Cumulative Layout Shift (CLS) changed slightly, from 0 seconds to 0.004 seconds.
- Speed Index decreased by 95,2%, from 6.3 seconds to 0.3 seconds.

1. First Contentful Paint (FCP): A decrease of 75% indicates a significant increase in the speed of displaying the first content on the page, which makes the application more attractive to users.
2. Largest Contentful Paint (LCP): The 92% reduction shows a dramatic improvement in the load time of the largest content on the page, which is immediately noticed by users.
3. Total Blocking Time (TBT): The reduction from 570 milliseconds to 0 milliseconds emphasizes the absence of blocking operations that can interfere with user interaction.
4. Cumulative Layout Shift (CLS): A slight change from 0 seconds to 0.004 seconds indicates a stable page display during loading.
5. Speed Index: A reduction of 95,2% indicates a significant acceleration of page rendering and contributes to an excellent user experience.

After analyzing a baseline web application and an application with recommendations implemented using Lighthouse, you can compare the resulting metrics and understand how the optimization techniques improved the performance, availability, compliance with best practices, and SEO of the web application. The results of the comparison will help identify the strengths and weaknesses of the application and direct efforts to further optimization and performance improvement.

6. Conclusions

As a result of the work carried out, which used modern open-source tools such as Lighthouse and K6 to optimize the web application based on Next.js, TypeScript, TRPC, Prisma, Postgres and Vercel, significant improvements in quality and application performance.

Analysis of the test results with K6 indicates a significant improvement in important metrics such as the quantity of HTTP requests, the duration of HTTP requests, the waiting time of HTTP requests, and the quantity of HTTP requests per second. This indicates an effective optimization that reduced the load on the server and made the application more responsive.

The results of the Lighthouse analysis are also impressive: reductions in First Contentful Paint, Largest Contentful Paint, Total Blocking Time, Cumulative Layout Shift and Speed Index indicate a significant acceleration of loading and displaying page content. This makes the app more attractive to users and improves their overall experience.

In summary, the use of optimization methods and performance measurement tools allowed to improve the quality and speed of the web application, ensuring user satisfaction and increasing its competitiveness in the market.

7. References

[1] Dudnik, Y. Kravchenko, O. Trush, O. Leshchenko, N. Dakhno and V. Rakytskyi, "Study of the Features of Ensuring Quality Indicators in Multiservice Networks of the Wi-Fi Standard," 2021 IEEE 3rd International Conference on Advanced Trends in Information Theory (ATIT), 2021, pp. 93-98, doi: 10.1109/ATIT54053.2021.9678691.

[2] Kravchenko, Y., Leshchenko, O., Dakhno, N., & Radko, M. (2022). Comparative evaluation of a universities' websites quality. development, 6, 7.

[3] Zimmermann, H. (1980). OSI reference model-the ISO model of architecture for open systems interconnection. IEEE Transactions on communications, 28(4), 425-432.

[4] N. Dakhno, O. Barabash, H. Shevchenko, O. Leshchenko and A. Musienko, "Modified Gradient Method for K-positive Operator Models for Unmanned Aerial Vehicle Control," 2020 IEEE 6th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), KYIV, Ukraine, 2020, pp. 81-84, doi: 10.1109/MSNMC50359.2020.9255516.

[5] Dakhno N., Barabash O., Shevchenko H., Leshchenko O., Dudnik A. Integro-differential Models with a K-symmetric Operator for Controlling Unmanned Aerial Vehicles Using a Improved Gradient Method. 2021 IEEE 6th International Conference "Actual Problems of Unmanned Aerial

Vehicles Development (APUAVD). Proceedings. October 19 – 21, 2021, Kyiv, Ukraine. P. 61 – 65. DOI: 10.1109/APUAVD53804.2021.9615431.

[6] Trush, O., Dudnik, A., Trush, M., Leshchenko, O., Shmat, K., & Mykolaichuk, R. (2022, December). Mask Mode Monitoring Systems Using IT Technologies. In 2022 IEEE 4th International Conference on Advanced Trends in Information Theory (ATIT) (pp. 219-224). IEEE. DOI: 10.1109/ATIT58178.2022.10024216

[7] https://lighthouse-metrics.com/lighthouse/checks/be00382f-301e-47e4-84c8-d0142b4ff870

[8] Butkiewicz, M., Madhyastha, H. V., & Sekar, V. (2013). Characterizing web page complexity and its impact. IEEE/Acm Transactions On Networking, 22(3), 943-956.

[9] Smith, P. G. (2012). Professional website performance: optimizing the front-end and back-end. John Wiley & Sons.

[10] Grigorik, I. (2013). High Performance Browser Networking: What every web developer should know about networking and web performance. " O'Reilly Media, Inc.".

[11] Kleppmann, M. (2017). Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. " O'Reilly Media, Inc.".

[12] Barker, T. (2014). High performance responsive design: Building faster sites across devices. " O'Reilly Media, Inc.".

[13] Z. Qian, H. Miao and H. Zeng, "A Practical Web Testing Model for Web Application Testing," 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, Shanghai, China, 2007, pp. 434-441, doi: 10.1109/SITIS.2007.16.

[14] Saputro, P. H. (2023). Application of GTMetrix and K6 in Performance Testing and Stress Levels on POS (Point Of Sale) Websites (Case Study on waroeng99 Website). Jurnal Sistem Informasi dan Teknologi Informasi, 2(1), 1-11.

[15] Wang, P., Varvello, M., & Kuzmanovic, A. (2019, July). Kaleidoscope: A crowdsourcing testing tool for web quality of experience. In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS) (pp. 1971-1982). IEEE.

[16] Kaur, S., Kaur, K., & Kaur, P. (2016). An empirical performance evaluation of universities website. International Journal of Computer Applications, 146(15), 10-16.

[17] URL: https://yslow.org/

[18] Matam, S., & Jain, J. (2017). Pro Apache JMeter: web application performance testing. Apress.

[19] Heričko, T., Šumak, B., & Brdnik, S. (2021, September). Towards Representative Web Performance Measurements with Google Lighthouse. In Proceedings of the 2021 7th Student Computer Science Research Conference (p. 39).

[20] James, I. (2019). Webwaves: Web page auditing using Lighthouse. Preview, 2019(203), 50-51.

[21] Patil, S. A. (2020). Comparative SEO Techniques Analysis on core WebPages and its Effectiveness in Context of Google Search Engine, International Journal of Scientific Development and Research, Vol 5, Is.3 (pp. 420-428)

[22] Rosenfeld, L. & Morville, P. & Arango, J. (2015). Information Architecture: For the Web and Beyond, O'Reilly Media

[23] Vasilijević, V., Kojić, N., & Vugdelija, N. (2020, October). A new approach in quantifying user experience in web-oriented applications. In 4th International Scientific Conference on Recent Advances in Information Technology, Tourism, Economics, Management and Agriculture–ITEMA (pp. 9-16).

[24] Z. Qian, H. Miao and H. Zeng, "A Practical Web Testing Model for Web Application Testing," 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, Shanghai, China, 2007, pp. 434-441, doi: 10.1109/SITIS.2007.16.

[25] Akpinar, P., Aktas, M. S., Keles, A. B., Balaman, Y., Guler, Z. O., & Kalipsiz, O. (2020, June). Web application testing with model based testing method: case study. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)* (pp. 1-6). IEEE.