Optimizing and Improvement a Web Application Using Open Source Tools

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

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 application. 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
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...
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 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:

   \[
   \text{Response Time} = N \sum T, \tag{1}
   \]

   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.

   \[
   \text{Page Load Time} = \sum \text{Resource Load Time}, \tag{2}
   \]

   where \(\text{Resource Load Time}\) - 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.

   \[
   \text{CPU Usage} = \frac{\text{Used CPU Time}}{\text{Total CPU Time}} \times 100\%, \tag{3}
   \]

   \[
   \text{Memory Usage} = \frac{\text{Used Memory}}{\text{Total Memory}} \times 100\%, \tag{3}
   \]

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

   \[
   \text{Reliability} = \frac{\text{Number of Errors}}{\text{Total Interactions}} \times 100\%, \tag{4}
   \]

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.

   \[
   \text{User Interaction Response Time} = \text{EndTime} - \text{StartTime}, \tag{5}
   \]

   where \(\text{EndTime}\) - is the time of receiving a response, and \(\text{StartTime}\) - 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

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.

3. **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.

4. **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.

![Figure 2: Example of using GTmetrix](image1)

![Figure 3: Using WebPageTest](image2)
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.

Figure 4: Pingdom usage

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.

![Figure 6: Indicator and its weight in efficiency calculation](image)

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:

\[
FCP = t_{FCP} - t_{start},
\]

where:
- \(t_{FCP}\) - the time when the first content becomes visible.
- \(t_{start}\) - the time the page started to load.

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:

\[
CLS = \sum (impact\ fraction \ast distance\ fraction \ast distance),
\]

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.

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:

\[
LCP = t_{LCP} - t_{start},
\]

where:
- \(t_{LCP}\) - the time when the largest content becomes visible.
- \(t_{start}\) - the time the page started to load.

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:

\[
TBT = \sum BT_i, \text{де } i \text{ від } 1 \text{ до } N,
\]

where:
- \(BT_i\) - 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.
- \(N\) - Quantity of tasks that blocked the page.
- Total Blocking Time, TBT: This is the total block time for all tasks.

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.

<table>
<thead>
<tr>
<th>Speed Index (in seconds)</th>
<th>Color-coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–3.4</td>
<td>Green (fast)</td>
</tr>
<tr>
<td>3.4–5.8</td>
<td>Orange (moderate)</td>
</tr>
<tr>
<td>Over 5.8</td>
<td>Red (slow)</td>
</tr>
</tbody>
</table>

**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.

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

**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.

```javascript
export const options = {
  stages: [
    { duration: "10s", target: 100 }, // Наvantаження 100 віртуальних користувачів протягом 10 секунд
    { duration: "1m", target: 100 }, // Утримання наvantаження 100 віртуальних користувачів протягом наступнух 60 секунд
    { duration: "10s", target: 0 }, // Зниження наvantаження до 0 протягом 10 секунд
  ],
  thresholds: {
    http_req_duration: "("95%<500")", // Установлення порогу, до якого запити 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, 312.17 ms 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.
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.

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


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


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


