УДК 004.05:021
DOI https://doi.org/10.32689/maup.it.2023.1.2

Надія БОЛЮБАШ
кандидат педагогічних наук, доцент, доцент кафедри інтелектуальних інформаційних систем, Чорноморський національний університет імені Петра Могили, вул. 68 Десантників, 10, Миколаїв, Україна, індекс 54003 (Nadiya.Bolubash@chmnu.edu.ua)

ORCID: 0000-0002-2274-2422

Михайло ОЛІЙНИК
магістрант, кафедра інтелектуальних інформаційних систем, Чорноморський національний університет імені Петра Могили, вул. 68 Десантників, 10, Миколаїв, Україна, індекс 54003 (mischa.oleinik2016@gmail.com)

ORCID: 0009-0007-3144-6725

Nadiia BOLIUBASH
PhD in Education, Associate Professor, Associate Professor at the Department of Intelligent Information Systems, Petro Mohyla Black Sea National University, 10 68 Desantnykiv str., Mykolaiv, Ukraine, postal code 54003 (Nadiya.Bolubash@chmnu.edu.ua)

Mykhailo OLIINYK
Master's Student at the Department of Intelligent Information Systems, Petro Mohyla Black Sea National University, 10 68 Desantnykiv str., Mykolaiv, Ukraine, postal code 54003 (mischa.oleinik2016@gmail.com)

Бібліографічний опис статті: Болюбаш, Н., Олійник, М. (2023). Методи підвищення продуктивності прогресивного вебзастосунку бібліотеки на основі моделі RAIL. *Інформаційні технології та суспільство*. Вип. 1 (7), 13–20. DOI: https://doi.org/10.32689/maup.it.2023.1.2

Bibliographic description of the article: Boliubash, N., Oliinyk, M. (2023). Metody pidvyshchennia produktyvnosti prohresyvnoho vebzastosunku biblioteky na osnovi modeli RAIL [Methods for increasing the performance of the library's progressive web application based om the RAIL model]. *Informatsiini tekhnolohii ta suspilstvo – Information technology and society*, 1 (7), 13–20. DOI: https://doi.org/10.32689/maup.it.2023.1.2

МЕТОДИ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ПРОГРЕСИВНОГО ВЕБЗАСТОСУНКУ БІБЛІОТЕКИ НА ОСНОВІ МОДЕЛІ RAIL

Анатація. *У статті розглянуто напрями цифрової трансформації бібліотечної сфери та обгрунтовано підходи, спрямовані на оптимізацію продуктивності електронних бібліотек.* ***Метою статті*** *є дослідження методів підвищення продуктивності прогресивного вебзастосунку бібліотеки у процесі його розробки з використанням орієнтованої на користувача моделі оптимізації продуктивності RAIL.* ***Методи дослідження.*** *Використано загальнонаукові методи аналізу та синтезу, методи розробки прогресивних вебзастосунків та методи оптимізації їх продуктивності, що базуються на оцінці архітектурних рішень з точки зору їх впливу на швидкість роботи застосунку.* ***Наукова новизна дослідження*** *полягає у виявленні в умовах сучасної цифрової трансформації бібліотечної сфери методів підвищення продуктивності вебзастосунку бібліотеки, адаптованого до роботи на різних апаратних та програмних платформах – прогресивного вебзастосунку, який поєднує у собі властивості вебзастосунку та нативного застосунку.* ***Висновки.*** *Використання сучасних цифрових систем обробки інформації вимагає перегляду підходів до вибору архітектур вебзастосунків, які бібліотеки використовують як платформи для взаємодії з читачами, та до визначення їх продуктивності. Для забезпечення кроспатформеності та можливості роботи із бібліотечними ресурсами у режимі offline виявлено доцільність розробки вебзастосунку бібліотеки з використанням тезнології PWA, архітектура якого включає маніфест застосунку, Service Workers та Application Shell. Дослідження методів підвищення продуктивності прогресивного вебзастосунку бібліотеки дозволило визначити стратегію застосуваня у процесі його розробки методів оптимізації продуктивності відповідно до вимог моделі RAIL: стиснення розміру зображень та задання їх розмірів явним способом через атрибути; зменшення JavaScript коду – оптимізуючи функції та розділивши тривалі завдання на завдання з тривалістю не більшою за 50 мс шляхом впровадження асинхронного коду; реалізація плавної анімації за допомогою CSS5 із обробкою кожного кадру до 10 мс; групування завдань, які виконуються у фоновому режимі, у блоки, не більші за 50 мс.*

Ключові слова: *цифрова бібліотека, технологія PWA, технологія Service Workers, прогресивний вебзастосунок, прогресивна вебметрика, модель RAIL.*

METHODS FOR INCREASING THE PERFORMANCE OF THE LIBRARY'S PROGRESSIVE WEB APPLICATION BASED ON THE RAIL MODEL

Abstract. *The article examines the directions of digital transformation of the library sphere and substantiates the approaches aimed at optimizing the productivity of electronic libraries.* ***The purpose*** *of the article is to study the methods of improving the performance of a progressive web application of the library during its development using the RAIL user-oriented performance optimization model.* ***Research methods.*** *General scientific methods of analysis and synthesis, methods of developing progressive web applications and methods of optimizing their performance, based on the assessment of architectural solutions from the point of view of their impact on the speed of the application, are used.* ***The scientific novelty*** *of the research consists in identifying, in the conditions of modern digital transformation of the library sphere, methods of increasing the productivity of the library web application adapted to work on various hardware and software platforms - a progressive web application that combines the properties of a web application and a native application.* ***Conclusions.*** *The use of modern digital information processing systems requires a review of approaches to the selection of web application architectures that libraries use as platforms for interacting with readers, and to determining their performance. To ensure cross-platform compatibility and the possibility of working with library resources in offline mode, the expediency of developing a library web application using PWA technology, whose architecture includes an Web App Manifest, Service Workers and Application Shell, was found. The study of methods for improving the performance of the progressive web application of the library allowed to determine the application strategy in the process of its development of performance optimization methods in accordance with the requirements of the RAIL model: compressing the size of images and specifying their dimensions in an explicit way through attributes; JavaScript code reduction – by optimizing functions and splitting long tasks into tasks with a duration of no more than 50ms by implementing asynchronous code; implementation of smooth animation using CSS5 with processing of each frame up to 10 ms; grouping tasks that are performed in the background into blocks of no more than 50 ms.*

Key words: *Digital Library, technology PWA, technology Service Workers, Progressive Web App, Progressive Web Metrics, model RAIL.*

Постановка проблеми та аналіз останніх досліджень і публікацій. Високі темпи інформатизації сучасного суспільства супроводжуються розширенням взаємодії через web-інтерфейс у всіх галузях оточуючої дійсності. У процесі цифрової трансформації бібліотечної сфери здійснюється перехід до електронних бібліотек, які стали одним із найбільш перспективних каналів доступу до бібліотечних ресурсів завдяки більш швидкому обслуговуванню читачів, полегшенню доступу до інформації та її пошуку, наявності онлайн послуг, можливості працювати з ресурсами онлайн та скачувати їх у різних форматах [7; 10; 11; 12]. Надання електронних онлайн послуг впроваджують у свою діяльність усі сучасні бібліотеки – академічні, вузькоспеціалізовані та публічні, призначені для широкого кола читачів. Зростання обсягів цих послуг потребує більш прогресивних підходів до розробки вебзастосунків, які використовуються у цифрових бібліотеках для взаємодії з читачами [4]. Одним із таких підходів, спрямованих на покращення якості обслуговування читачів та підвищення їх залучення, є реалізація послуг цифрової бібліотеки з застосуванням технології PWA.

В умовах широкого розповсюдження мобільного Інтернету і мобільних пристроїв ключем до успіху вебзастосунку є його продуктивність та кросплатформеність. Проте далеко не усі цифрові бібліотеки мають задовільні показники продуктивності електронних платформ й адаптовані до роботи на мобільних пристроях. Впровадження у їх діяльність прогресивних web-застосунків (англ. Progressive Web App, PWA) дозволяє забезпечити високі показники продуктивності при доступі до бібліотечних ресурсів із різних апаратних та програмних платформ [8].

У процесі розробки вебзастосунку бібліотеки важливим є вибір методів, спрямованих на підвищення продуктивності з метою забезпечення оптимальних показників. Значна частина методів, які дозволяють виявити проблемні місця, націлена на оцінку та оптимізацію продуктивності уже розробленого вебзастосунку. А метрики, які застосовують для оцінки продуктивності, здебільшого характеризують продуктивність з технологічної точки зору без урахування її суб'єктивного сприйняття користувачем [1; 2]. Підхід, реалізований у моделі RAIL (англ. Response Animation Idle Load), є більш ефективним, оскільки вона надає можливість вимірювати та оптимізувати показники продуктивності, орієнтовані на користувача, у процесі розробки із забезпеченням розширених можливостей доступності [5]. Проте методи підвищення продуктивності вебзастосунку відповідно до основних показників моделі RAIL є не достатньо дослідженими й потребують подальшого опрацювання.

Виклад основного матеріалу. Проведене дослідження дозволило установити, що прогресивні вебзастосунки є новим поколінням застосунків, які можуть адаптивно переналаштовуватися під параметри пристрою, на якому їх відкрито (смартфоні, планшеті, десктопному ПК, ноутбукові, нетбуку) та працювати в усіх операційних системах і браузерах таким чином, що робота з сайтом схожа на роботу з нативним застосунком [13]. За рахунок завчасного кешування даних за технологією Service Workers прогресивний web-застосунок бібліотеки дозволяє взаємодіяти з електронним ресурсом у режимі offline незалежно від з'єднання з Інтернетом, отримувати доступ до апаратних засобів пристрою та

відправляти push-повідомлення. Основні переваги PWA, які роблять їх ефективними для користувачів електронної бібліотеки, полягають у швидкому завантаженні, прогресивності, адаптивності та можливості працювати з бібліотечними ресурсами незалежно від з'єднання з Інтернетом [3].

Технологія PWA є більш зручною альтернативою технології DRM – Digital Rights Management, яка забезпечує управління доступом до цифрових даних з метою захисту авторських прав у випадку платного чи обмеженого доступу до бібліотечних ресурсів. Зазвичай DRM використовують для обмеження копіювання, друку та викладання у загальний доступ електронних книг. Використання технології PWA дозволяє вебзастосунку бібліотеки працювати в автономному режимі, пропонуючи читання у режимі offline без необхідності встановлення додаткових програм DRM, використовуючи дані, кешовані під час попередніх сеансів роботи із застосунком у фоновому режимі [11].

Для створення вебзастосунку бібліотеки було обрано технологію PWA, до базових складових якої відносять [8]:

1) маніфест застосунку (англ. Web App Manifest), який застосовують для надання застосунку нативних функцій, таких як іконка на робочому столі, ім'я, стартова сторінка, колір теми тощо;

2) технологію Service Workers, яка є основою PWA, включає перш за все обробку кешованих ресурсів для забезпечення роботи в автономному режимі, здійснює обробку запитів браузера, фонову синхронізацію та push-повідомлення.

3) ахітектуру Application Shell (оболонка застосунку), яка є оболонкою нативної програми, що зберігається на пристрої клієнта і за рахунок цього здійснює швидке завантаження з Service Workers.

Створення статичної оболонки користувальницького інтерфейсу PWA, яка кешується на пристрої користувача та забезпечує швидке завантаження, здійснено із використанням JavaScript та мови розмітки HTML5 і CSS3 як засобу стилізації. У процесі створення електронної бібліотеки було використано Google Books API, який дозволяє програмно виконувати операції в інтерактивному режимі на веб-сайті Google Books, що містить більше 10 млн книг у електронному форматі, безпосередньо у web-застосунку бібліотеки.

Для розробки PWA бібліотеки використано середовище розробки Web-Storm, фреймворк AngularJS та мову TypeScript, що компілюється в JavaScript. Бібліотека RxJS застосовувалася для написання запитів і back-end частини з опорою на методики реактивного програмування. Це дало можливість відслідковувати асинхронні потоки даних, які генерують події, упорядковані у часі, та відповідним чином реагувати на них.

Дослідження існуючих підходів до оптимізації продуктивності вебзастосунків дозволило установити, що під продуктивністю розуміють швидкість відгуку застосунку на запити та дії користувача [2]. Підвищення продуктивності вебзастосунку вимагає складних інженерних рішень, які передбачають застосування методів оптимізації на різних рівнях абстракції, до яких відносять кешування об'єктів із використанням HTTP заголовків, зменшення кількості HTTP-запитів, оптимізацію завантаження JavaScript коду, стиснення текстового контексту та зображень, зменшення кількості DNS-запитів, скорочення та об'єднання файлів CSS, JavaScript і HTML, управління перенаправленням між сторінками та оптимізацію доставки CSS [6; 9]. Робота прогресивних вебзастосунків здійснюється по протоколу HTTPS, забезпечуючи безпечну передачу даних між сервером і клієнтом та має свою специфіку, яка обумовлює необхідність розробки адаптованих до неї методів оптимізації.

Вимірювання продуктивності прогресивного web-застосунку є складним завданням, оскільки необхідно забезпечити оптимальні показники на різних апаратних та програмних платформах. З метою оцінювання продуктивності вебзастосунку в основному відслідковують такі події [5]:

1) DOMContentLoaded-подія, яка відбувається, коли сторінку – весь код HTML, завантажено та пройдено парсером не чекаючи завершення завантаження таблиць стилів, скрипти тільки почали виконуватися;

2) Load-подія, яка відбувається, коли сторінка повністю завантажена і відпрацювали всі скрипти, користувач може повноцінно взаємодіяти зі сторінкою.

Однак вказані метрики не завжди є оптимальними, час парсингу та виконання скриптів може бути занадто великим. Розробники разом із науковою спільнотою працюють над розробкою прогресивних вебметрик (англ. Progressive Web Metrics, PWM). Сукупність таких метрик, орієнтованих на користувача, містить модель RAIL, яка оцінює архітектурні рішення з точки зору їх впливу на швидкість роботи застосунку, установлюючи граничні значення для швидкості відгуку на дії користувача, затримки анімації, часу завантаження вмісту сторінки [5].

Застосування моделі RAIL при розробці прогресивного вебзастосунку бібліотеки дозволило задачу підвищення продуктивності впровадити у процес розробки. Модель RAIL передбачає наступний набір інструкцій для вимірювання продуктивності web-застосунку, виділяючи основні аспекти у його життєвому циклі (рис. 1):

1) відповідь (англ. Response) на виконання дій користувача повинна становити не більше, ніж 100 мілісекунд;

2) анімація (англ. Animation) повинна мати затримку не більшу, ніж 60 кадрів у секунду;

3) очікування (англ. Idle) – простій між діями користувача використовується у роботі застосунку для виконання відкладеної роботи (зокрема завантаження частин, які ще не були завантажені);

4) завантаження (англ. Load) повинно становити не більше 1–2 секунд, тому при звернені користувача до застосунку завантажується тільки його базова версія.

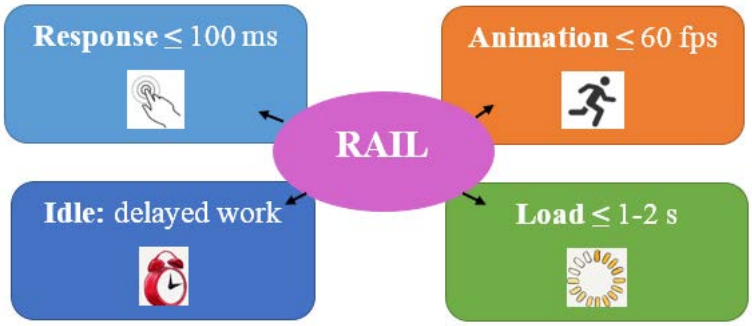

Рис. 1. Основні показники продуктивності моделі RAIL

Розроблений PWA бібліотеки є кросплатформенним та має розширену функціональність у порівнянні зі звичайними web-застосунками з доступом до пристроїв користувача. Дозволяє працювати в автономному режимі, забезпечує легкість установки, простоту налаштувань інтерфейсу, більшу ефективність, продуктивність та надійність навіть у локальній мережі. Вікно застосунку відкривається у браузері, однак на робочому столі є іконка, яка також дозволяє її відкрити. Користувач може виконати пошук бажаної книги, відфільтрувати книги по жанрам, переглянути список відібраних книг, більш детально ознайомитися з інформацією про окрему книгу та перейти до її читання онлайн якщо доступ до неї є безкоштовним. Є ресурси, доступ до яких є платним (рис. 2).

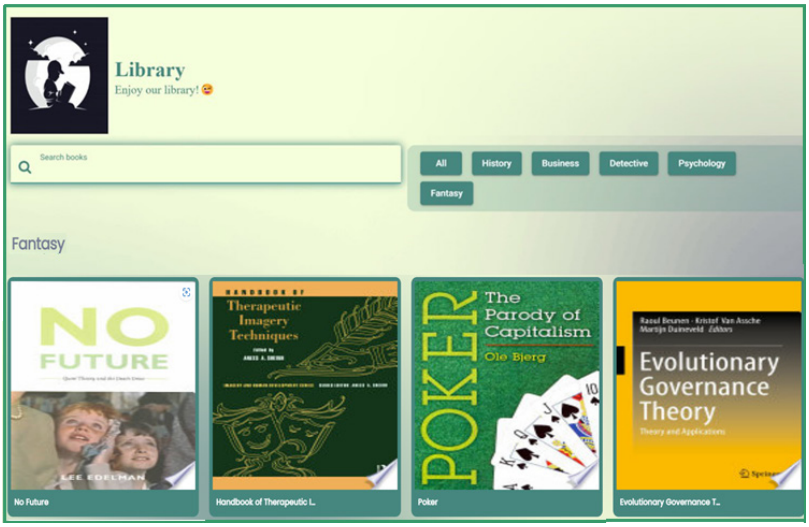

Рис. 2. Вікно створеного прогресивного вебзастосунку бібліотеки

У процесі розробки здійснювалася оцінка показників продуктивності вебзастосунку та їх оптимізація відповідно до моделі RAIL із використанням сервісів Chrome DevTools і Google PageSpeed Insights. Chrome DevTools, емулюючи різні пристрої, дозволяє адаптувати вебзастосунок під ПК, мобільні пристрої та екрани різних гаджетів. Це дає можливість оцінювати параметри продуктивності кросплатформенного вебзастосунку та здійснювати адаптивну верстку з CSS, HTML і JavaScript-кодом у процесі його розробки.

Lighthouse дозволяє діагностувати підтримку сукупності компонентів технології PWA у вебзастосунку: реєстрацію Servicer Workers, роботу в режимі offline, використання HTTPS та перенаправлення HTTP-трафіку в HTTPS та інших. У таблиці 1 відображено, як змінювалася оцінка відповідності стандартам PWA розроблюваного вебзастосунку бібліотеки по мірі додавання базових компонентів цієї

технології. Отримані результати свідчать про те, що розроблений вебзастосунок бібліотеки на різних платформах має високий ступінь відповідності стандартам технології PWA, якому відповідає оцінка в балах у межах від 90 до 100.

Таблиця 1

Зміна відповідності стандартам PWA вебзастосунку бібліотеки під час його розробки

Етап розробки	Ступінь відповідності PWA, *бали*	
	Десктопна версія	Мобільна версія
До впровадження базових компонентів PWA	40	36
Написання маніфесту	65	55
Впровадження Service Worker	94	91
Впровадження Application Shell	98	96

Google PageSpeed Insights з використанням технології Lighthouse оцінює продуктивність вебзастосунку на десктопній і мобільній версії, допомагає визначити, чим визвана низька продуктивність – проблемами у коді чи конфігурації сайту та надає рекомендації по поліпшенню продуктивності. Під час першого тестування оцінка продуктивності вебзастосунку бібліотеки показала, що є параметри, які не відповідали граничним значенням моделі RAIL.

Оптимальними з точки зору продуктивності для десктопної та мобільної версій виявилися наступні показники (табл. 2).

1. First Contentful Paint – час, необхідний браузеру для відображення першої частини вмісту DOM (текст, фонові зображення) становив 0,6 і 0,9 сек.

2. Speed Index – час візуального відображення контенту під час завантаження сторінки виявився рівним 0,7 та 0,8 секунд.

3. Total Blocking Time – час, протягом якого відбувається блокування сторінки у відповідь на дії користувача: кліки мишею або натискання клавіш становив 82 і 100 мс.

4. Cumulative Layout Shift – візуальна стабільність, яка оцінюється в балах від 0 до 1, виявилася близькою до нуля і була оцінена в 0,05 та 0,065 балів. Такі значення свідчать про незначне зміщення макета через асинхронне завантаження ресурсів.

Таблиця 2

Показники продуктивності до оптимізації часу завантаження Load

Показники продуктивності	Десктопна версія	Мобільна версія
First Contentful Paint	0,6 сек	0,9
Speed Index	0,7 сек	0,8
Largest Contentful Paint	4,5 сек	5,6
Time to Interactive	3,1 сек	3,9
Total Blocking Time	82 мс	100
Cumulative Layout Shift	0,05	0,065

Показники, які не відповідали моделі RAIL та потребували коригування:

1) Largest Contentful Paint – час завантаження найбільшого елемента в області перегляду становив 4,5 і 5,6 секунд для десктопної та мобільної версій відповідно;

2) Time to Interactive – час, за який сторінка стане повністю інтерактивною, виявився рівним 3,1 для десктопної версії та 3,9 секунд для мобільної версії.

Google PageSpeed Insights дозволив виявити, що основний час завантаження прийшовся на графічні зображення, зокрема – фотографію фону.

Для налаштування часу завантаження Load із метою його прискорення було зменшено розмір зображень із використанням сервісу TinyPNG та задано їх розміри явним способом через атрибути. Розмір фонового зображення було зменшено майже на 80%. Застосування цих підходів дозволило зменшити час завантаження найбільшого елемента в області перегляду у 4 рази – значення Largest Contentful Paintіз, змінилося із 4,5 до 1 секунди та із 5,6 до 1,4 секунди для для десктопної версії мобільної версії. Час, за який сторінка стає повністю інтерактивно, зменшився у 3 рази – значення показника Time to Interactive змінилося із 3,1 до 1,1 секунди для десктопної версії та із 3,9 до 1,3 секунди для мобільної версії (табл. 3). Нові результати оцінки часу завантаження Load цілком задовольняють критерії моделі RAIL.

Таблиця 3

Показники продуктивності після оптимізації часу завантаження Load

Показники продуктивності	Десктопна версія	Мобільна версія
First Contentful Paint	0,6 сек	0,8
Speed Index	0,6 сек	0,7
Largest Contentful Paint	1,0 сек	1,4
Time to Interactive	1,1 сек	1,3
Total Blocking Time	78 мс	96
Cumulative Layout Shift	0,051	0,062

Для оцінки швидкості відгуку вебзастосунку Response на дії користувача у процесі його взаємодії із інтерфейсом (клацання мишею, натискання клавіш клавіатури тощо) було використано сервіс Chrome DevTools, який дозволяє перевіряти мережевий трафік, швидкодію сайту та інші показники продуктивності на різних апаратних платформах (рис. 3). Було виявлено, що швидкість відгуку на дії користувача вебзастосунку електронної бібліотеки становила від 72 до 128 мілісекунд. Значення, більші за 100 мілісекунд, не відповідали вимогам моделі RAIL. Для прискорення відповіді Response вебзастосунку на дії користувача було здійснено зменшення JavaScript коду – оптимізацію функцій та розділення тривалих завдань шляхом впровадження асинхронного коду. Тривалим є таке завдання, виконання якого займає більше 50 мс. Після внесених змін час виконання тривалих завдання було зменшено більше ніж вдвічі, а швидкість відгуку Response не перевищувала 100 мс.

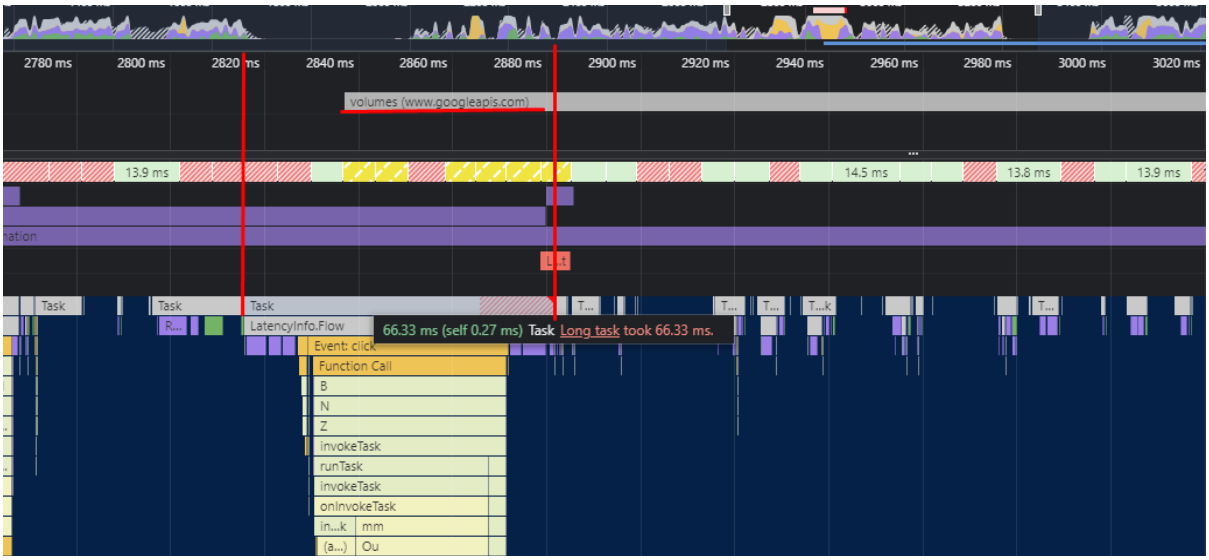

Рис. 3. Оцінка показників продуктивності вебзастосунку бібліотеки із використанням Chrome DevTools

Для перевірки показника Animation також було використано Chrome DevTools. Під анімацією, яка повинна бути неперервною, мають на увазі такі операції як скролінг, випадаючі списки, використання полос прокрутки та інші ефекти, пов'язані з тим, що вміст екрана повинен змінюватися протягом якогось часу. Chrome DevTools показує, скільки часу був активним той чи інший кадр.

Для правильного налаштування анімації відповідно до моделі RAIL кожен кадр повинен бути завершений за 16 мс (1/60 сек = 0,0166 мс). Проте оскільки браузеру потрібно близько 6 мс для рендерингу кожного кадру, кадр потрібно створити за 10 мс. Якщо час створення буде більшим, кадр буде пропущено а картинка на екрані буде перервною. У вебзастосунку бібліотеки анімацію було реалізовано з використанням CSS. При її перевірці на різних платформах не було виявлено проблем, анімація працювала плавно.

Відповідно до вимог моделі RAIL у той час, коли користувач не взаємодіє із інтерфейсом вебзастосунку, може бути виконана відкладена робота (англ. Idle Task) – ініціалізація різних компонентів, пошук і сортування даних, завантаження відкладених раніше елементів контенту, відправлення даних до сервісу аналітики тощо. При роботі з цифровою бібліотекою простій сторінки може бути у той час, коли користувач читає текст на екрані, не виконуючи ніяких дій. У цей час у фоновому режимі можуть заватажуватися ще не вікриті сторінки книги, із якою працює читач. Це дає можливість у подальшому отримувати доступ до сторінок книги навіть без підключення до мережі Інтернет.

Для правильного налаштування очікування Idle вебзастосунку – його роботи під час простою, було здійснено групування завдань, які виконуються у фоновому режимі, у блоки, не більші 50 мілісекунд (рис. 4). Це гарантує можливість відповіді на дії користувача у вікні відгуку 100 мілісекунд. Якщо користувач взаємодіє із сторінкою під час простою, взаємодія має вищий пріоритет і перериває фонову роботу під час простою.

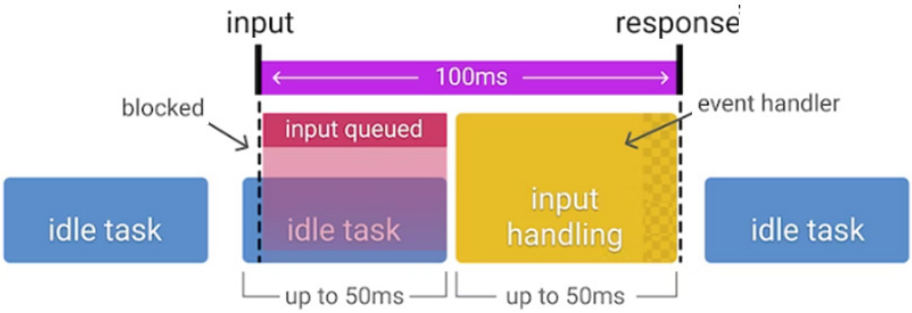

Рис. 4. Налаштування очікування Idle

Після здійснених налаштувань показники продуктивності вебзастосунку бібліотеки було оптимізовано у відповідності до вимог моделі RAIL Створений PWA бібліотеки вирішує такі проблеми, як миттєвий відруг на дії користувача, плавну анімацію та інтерактивність, швидкий доступ до ресурсів на різних апаратних та програмних платформах навіть при низькій швидкості Інтернету.

Висновки. Підвищення продуктивності прогресивного вебзастосунку бібілотеки вимагає глибокого розуміння конкретних сценаріїв його роботи та ретельного планування під час розробки послідовності застосування методів, спрямованих на оптимізацію його продуктивності.

Виявлено наступні методи, спрямовані на підвищення продуктивності вебзастосунку відповідно до моделі RAIL: 1) для зменшення часу завантаження Load – стиснення розміру зображень та задання їх розмірів явним способом через атрибути; 2) для зменшення часу відгуку на дії користувача Response – оптимізація функцій та розділення тривалих завдань шляхом впровадження асинхронного коду на завдання, тривалість яких не перевищує 50 мс; 3) для налаштування показника Animation – реалізація анімації за допомогою CSS, спрямована на забезпечення створення кожного кадру не більше ніж за 10 мс; 4) для налаштування очікування Idle – групування завдань, які виконуються у фоновому режимі, у блоки, не більші за 50 мс. Застосування вказаних методів дозволило майже вдвічі скоротити час відгуку Response, втричі – час завантаження Load, забезпечити плавну роботу анімації на пристроях із різною частотою оновлення екрану та виконання відкладеної роботи у фоновому режимі.

Розроблений прогресивний вебзастосунок надає доступ до ресурсів Google Books. Базуючись на виявлених методах підвищення продуктивності, які відповідають вимогам орієнтованої на користувача моделі RAIL, може бути здійснена розробка прогресивного вебзастосунку бібліотеки, адаптована до цифрових ресурсів іншої бібліотеки. Що дасть можливість користувачам отримувати доступ до її ресурсів із високою продуктивністю на різних апаратних та програмних платформах у онлайн та offline режимах незалежно від наявності зв'язку з Інтернетом.

Список використаних джерел:

1. Гридин В.Н., Анисимов В.И., Васильев С.А. Методы повышения производительности современных веб-приложений. *Известия ЮФУ. Технические науки*. № 2(212). 2020. С. 193-200. https://doi.org/10.18522/2311-3103-2020-2-193-200.

2. Максимов А.Я., Мартышкин А.И. Обзор современных програмных решений в области измерения производительности клиентской части веб-приложений. *Современные наукоемкие технологии*. № 12(2). 2021. С. 348-354. https://doi.org/10.17513/snt.39001.

3. Ткачук В. PWA, як перспективний напрямок об'єднання мобільних технологій. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. Вип. 46. 2022. С. 83-87. https://doi.org/10.36910/6775-2524-0560-2022-46-12.

4. Gaffar S.A., Kishore Kumar Dr.S. Awareness and access to mobile applications in an Academic Library. *Library Philosophy and Practice* (e-journal). 2019. URL: https://digitalcommons.unl.edu/libphilprac.

5. Irish P., Lewis P. Introducing RAIL: a user-centric model for performance. 2015. URL: https://www.smashingmagazine.com/2015/10/rail-user-centric-model-performance/.

6. Jasper R., Malavolta I., Taher A. Optimize along the way: An industrial case study on web performance. *Journal of Systems and Software*. Vol. 198. 2022. https://doi.org/10.1016/j.jss.2022.111593.

7. Kari H.K. Digital Transformation of Information and its Impact on Libraries. *World Journal of Innovative Research*, 9(1), P. 26-30. 2020. URL: https://www.wjir.org/download_data/WJIR0901033.pdf.

8. Majchrzak T.A., Andreas B.H., Grønli T.M. Progressive Web App: the Definite Approach to Cross-Platform Development? In *Processings of the 51st Hawaii International Conference on System Sciences*. 2018. pp. 5735-5744. https://doi.org/10.24251/hicss.2018.718.

9. Malavolta I. Beyond native apps: web technologies to the rescue. In: *Proceedings of the 1st International Workshop on Mobile Development*. 2016. P. 1–2. https://doi.org/10.1145/3001854.3001863.

10. Panda S. Digital Rights Management (DRM) in the Libraries of Digital-era: Concepts, IPR Issues & Concerns of LIS Community. *Library Philosophy and Practice* (e-journal). 2021. URL: https://digitalcommons.unl.edu/libphilprac/6645.

11. Panda S.A. Study of On-the-Go Reference Service Using Mobile Technology in Library. In *Re-Envisioning Roles and Responsibilities of Library Professionals in the New Normal*. Daryaganj, New Delhi, India: DPS Publishing House, 2021. P. 83-99. http://doi.org/10.5281/zenodo.5091312.

12. Rahane V.C. Mobile Technology using in Library services. *International Journal of Research in Library Science*. Vol. 4(2). 2018. P. 23-26. URL: https://doi.org/10.26761/ijrls.4.2.2018.1294.

13. Sheppard D. Beginning Progressive Web App Development: Creating a Native App Experience on the Web. CA: Pub. Apress Bercel. 2017. P. 266. http://doi.org/10.1007/978-1-4842-3090-9.

References:

1. Gridin V.N., Anisimov V.I., Vasil'ev S.A. Metody povysheniya proizvoditel'nosti sovremennyh veb-prilozhenij [Performance Techniques for Modern Web Applications]. *Izvestiya YUFU. Tekhnicheskie nauki – Izvestiya SFedU. Technical science*. № 2(212). 2020. P. 193-200. https://doi.org/10.18522/2311-3103-2020-2-193-200.

2. Maksimov A.YA., Martyshkin A.I. Obzor sovremennyh programnyh reshenij v oblasti izmereniya proizvoditel'nosti klientskoj chasti veb-prilozhenij [Overview of modern software solutions in the field of measuring the performance of the client side of web applications] *Sovremennye naukoemkie tekhnologii – Modern high technologies*. № 12(2). 2021. P. 348-354. https://doi.org/10.17513/snt.39001.

3. Tkachuk V. PWA, yak perspektyvnyi napriamok obiednannia mobilnykh tekhnolohii [PWA, as a promising direction of combining mobile technologies] *Kompiuterno-intehrovani tekhnolohii: osvita, nauka, vyrobnytstvo – Computer-integrated technologies: education, science, production*. № 46. 2022. P. 83-87. https://doi.org/10.36910/6775-2524-0560-2022-46-12.

4. Gaffar S.A., Kishore Kumar Dr.S. Awareness and access to mobile applications in an Academic Library. *Library Philosophy and Practice* (e-journal). 2019. URL: https://digitalcommons.unl.edu/libphilprac.

5. Irish P., Lewis P. Introducing RAIL: a user-centric model for performance. 2015. URL: https://www.smashingmagazine.com/2015/10/rail-user-centric-model-performance/.

6. Jasper R., Malavolta I., Taher A. Optimize along the way: An industrial case study on web performance. *Journal of Systems and Software*. Vol. 198. 2022. https://doi.org/10.1016/j.jss.2022.111593.

7. Kari H.K. Digital Transformation of Information and its Impact on Libraries. *World Journal of Innovative Research*, 9(1), P. 26-30. 2020. URL: https://www.wjir.org/download_data/WJIR0901033.pdf.

8. Majchrzak T.A., Andreas B.H., Grønli T.M. Progressive Web App: the Definite Approach to Cross-Platform Development? In *Processings of the 51st Hawaii International Conference on System Sciences*. 2018. pp. 5735-5744. https://doi.org/10.24251/hicss.2018.718.

9. Malavolta I. Beyond native apps: web technologies to the rescue. In: *Proceedings of the 1st International Workshop on Mobile Development*. 2016. P. 1–2. https://doi.org/10.1145/3001854.3001863.

10. Panda S. Digital Rights Management (DRM) in the Libraries of Digital-era: Concepts, IPR Issues & Concerns of LIS Community. *Library Philosophy and Practice* (e-journal). 2021. URL: https://digitalcommons.unl.edu/libphilprac/6645.

11. Panda S.A. Study of On-the-Go Reference Service Using Mobile Technology in Library. In *Re-Envisioning Roles and Responsibilities of Library Professionals in the New Normal*. Daryaganj, New Delhi, India: DPS Publishing House, 2021. P. 83-99. http://doi.org/10.5281/zenodo.5091312.

12. Rahane V.C. Mobile Technology using in Library services. *International Journal of Research in Library Science*. Vol. 4(2). 2018. P. 23-26. URL: https://doi.org/10.26761/ijrls.4.2.2018.1294.

13. Sheppard D. Beginning Progressive Web App Development: Creating a Native App Experience on the Web. CA: Pub. Apress Bercel. 2017. P. 266. http://doi.org/10.1007/978-1-4842-3090-9.