УДК 004.021

ОБЗОР СОВРЕМЕННЫХ ПРОГРАММНЫХ РЕШЕНИЙ В ОБЛАСТИ ИЗМЕРЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЙ

Максимов Я.А., Мартышкин А.И.

ФГБОУ ВО «Пензенский государственный технологический университет», Пенза, e-mail: Starspeen@gmail.com, Alexey314@yandex.ru

В данной статье предлагается обзор популярных современных программных решений в области измерения производительности клиентской части веб-приложений, их достоинства и недостатки. Некоторые из перечисленных решений могут быть использованы не только на клиентской, но и на серверной части веб-приложений, но в рамках данной статьи эти программы рассматриваются как инструменты для проведения тестирования клиентских веб-приложений. В статье подчеркнута важность и актуальность данной темы, так как инструменты и подходы для разработки клиентских веб-приложений эволюционируют в современном вебе настолько быстро, что даже разработчики инструментов для оценки производительности не успевают подстраиваться под такие новые технологии, кроме того, в настоящее время нет четких рамок того, какие значения для определенных метрик производительности являются оптимальными. Цель статьи заключается в рассмотрении особенностей проблем, с которыми могут сталкиваться разработчики при измерении производительности клиентской части веб-приложений. Перечисленные проблемы могут послужить руководством для других разработчиков при тестировании подобных приложений, а также стать фундаментом для разработки системы для измерения производительности клиентской части веб-приложений, учитывающей все перечисленные минусы других программных решений.

Ключевые слова: оптимизация веб-приложений, клиентские веб-приложения, производительность, браузер, интернет, пользовательский интерфейс

OVERVIEW OF MODERN SOFTWARE SOLUTIONS IN THE AREA OF MEASURING THE PERFORMANCE OF THE CLIENT PART OF WEB- APPLICATIONS

Maksimov Ya.A., Martyshkin A.I.

Penza State Technological University, Penza, e-mail: Starspeen@gmail.com, Alexey314@yandex.ru

This article provides an overview of popular modern software solutions in the field of measuring the performance of the client side of web applications, their advantages, and disadvantages. Some of the listed solutions can be used not only on the client side, but also on the server side of web applications, but, within the framework of this article, these programs are considered as tools for testing client web applications. The article emphasizes the importance and relevance of this topic, since tools and approaches for developing client-side web applications are evolving on the modern web at a tremendous speed, that even the developers of performance assessment tools do not have time to adapt to such new technologies, in addition, at the present time there is not a clear framework for what values are optimal for certain performance metrics. The purpose of the article is to consider the features of all the current problems that developers may encounter when measuring the performance of the client side of web applications. The listed problems can be helpful as a guide for other developers when they are testing similar applications, also it can be the foundation for developing a system for measuring the performance of the client side of web applications, considering all the listed disadvantages of other software solutions.

Keywords: web application optimization, client web applications, performance, browser, internet, user interface

С момента появления интернета прошло около шестидесяти лет, и за это время он стал хранилищем многочисленной информации, местом размещения сайтов или веб-приложений, предназначенных для развлекательной или научной сфер деятельности. С каждым годом количество пользователей интернета растет, и также параллельно с этим увеличивается количество веб-приложений, которые используются людьми для совершения определенных операций [1]. Обычно при работе с ними пользователи взаимодействуют, в частности, с клиентской частью, и им важно получить стабильно работающее приложение без ошибок и «тормозов». В связи с большим количеством спроса на такие технологии изменились и требования к их производительности и отзывчивости со стороны пользователей. Это происходило по мере того, как веб-сайты все более наполнялись дополнительным функционалом, различными абстракциями в кодовой базе и интерактивными анимациями. В то же время производительность являлась ключом к обеспечению пользователю отзывчивого приложения, как с точки зрения быстроты загрузки начальной страницы, так и своевременного реагирования при взаимодействии пользователя, с элементами пользовательского интерфейса с минимально возможным ожиданием для него.

В современных реалиях создание пользовательских интерфейсов в веб-приложениях является чрезвычайно сложной задачей. Разработчики должны учитывать много факторов, таких как доступность, производительность, изменение требований к проекту, адаптирование пользовательского интерфейса под различные типы устройств и браузеров и многое другое. В настоящее время есть множество фреймворков и инструментов, которые помогают делать работу архитекторов программного обеспечения проще. Темпы эволюции в текущей среде фреймворков для *JavaScript* высоки. Так, например, существующие фреймворки быстро развиваются, в то же время новые фреймворки появляются на свет каждый год.

Исходя из вышеописанного, цель статьи заключается в рассмотрении особенностей актуальных проблем, с которыми могут сталкиваться разработчики при измерении производительности клиентской части веб-приложений. Перечисленные проблемы могут послужить руководством для других разработчиков при тестировании подобных приложений, а также стать фундаментом для разработки системы для измерения производительности клиентской части веб-приложений, учитывающей все перечисленные минусы других программных решений.

Материалы и методы исследования

Стандарт *IEEE 610,* определяющий термины, используемые в настоящее время в компьютерной сфере и устанавливающий стандартные определения этих терминов, дает следующее определение термину производительность: «Производительность – это степень, в которой система или компонент выполняет свои назначенные функции в рамках заданных ограничений, таких как скорость, точность или использование памяти» [2]. Тест производительности или бенчмарк – это итоговая задача, необходимая для определения сравнительных характеристик производительности компьютерной системы. Иногда бенчмарками также называются программы, которые тестируют время автономной работы ноутбуков и карманных персональных компьютеров, радиус действия беспроводной сети, пропускную способность каналов передачи данных, амплитудно-частотную характеристику звукового тракта и другие доступные для измерения характеристики, напрямую не связанные с производительностью [3]. Тесты производительности особо важны для разработчиков веб-приложений, будь то клиентская или серверная часть, потому что в последнее десятилетие многие компании перенесли свои сервисы в веб. Это связано с тем, что пользователи могут в любой момент из любой точки земного шара [4] и с любого устройства получить доступ к файлам в облаке или к почте.

Простые сайты во время своей работы не требовательны к ресурсам компьютера, но сложные корпоративные клиентские веб-приложения, или *Single Page Application,* состоят из большого количества элементов и логики, которая реализуется на языке программирования *JavaScript,* и в таких случаях скорость работы может быть неоптимальной. Пользователям будет неудобно взаимодействовать с такой системой из-за большого количества времени, которое затрачивается на первую отрисовку всей страницы или долгого отклика системы на действия пользователя. Как правило, корень проблемы заключается в том, что перед разворачиванием разработанной клиентской части веб-приложения разработчики не тестируют скорость работы своего детища при различных пользовательских вариантах взаимодействия с системой или не берут в расчет, что ей будут пользоваться люди с более слабыми компьютерами / мобильными устройствами.

Проблема производительности клиентских веб-приложений всегда была популярна, потому что перед разработчиками таких приложений имеется огромное количество неизвестных. Так, например, для разных приложений нет конкретных стандартов, описывающих, что считается хорошим показателем производительности (для времени отрисовки, для выделяемой памяти приложения), а что плохим. Частично эту проблему решают метрики *Web Vitals*, которые используются в поисковом движке компания *Google* [5]. *Web Vitals* – это набор конкретных метрик, которые *Google* считает важными для общего взаимодействия с пользователем веб-страницы. Впервые данные метрики были анонсированы 10 ноября 2020 г. [6]. Хотя они и являются популярными и многие разработчики используют их для того, чтобы оценить, готово ли их приложение для публикации конечному пользователю или нет, данные метрики не являются волшебным инструментом, применимым ко всем проектам. Не следует, однако, забывать о том, что в случае использования распределенной системы клиент-сервер, между узлами сети возникает очень нестабильное соединение, которое необходимо учитывать при измерении, а также что исполнение клиентского кода является не всегда очевидным в связи с тем, что такая среда исполнения, как браузер, считается непредсказуемой. Первопричиной этого является разная механика внутри браузеров,

и, более того, на него могут влиять и дополнения, установленные в браузер.

В современном вебе клиентские веб-приложения представляют из себя *Single Page Application.* Преимущество этой технологии заключается в том, что приложения такого вида во время работы дают пользователю возможность работать на одной странице в браузере, не требуя при этом перезагрузки страниц каждый раз, когда он переходит на новую страницу. В связи с тем, что страница загружается только один раз, производительность и удобство использования повышается, и браузеру не приходится загружать новую каждый раз. Это можно считать удобством с точки зрения пользователя, но с точки зрения тестирования производительности это добавляет проблемы, поскольку приложение не переходит на новый унифицированный указатель ресурса (или *URL)* каждый раз при взаимодействии с пользователем. По данной причине необходимо использовать инструментарий, который может воспроизводить сценарии действий пользователя в браузере, а затем использовать их для проведения тестирования производительности. Для оценки состояния производительности у современных клиентских веб-приложений существуют различные браузерные расширения, *desktop*-приложения, веб-приложения, которые в своих алгоритмах используют уникальные методы обработки данных страницы для выдачи результатов о производительности. Несмотря на то, что *Single Page Application* представляют собой относительно новую технологию, которая приносит разработчикам множество возможностей для разработки клиентских веб-приложений, но на самом деле используются все те же ресурсы и подходы, как и для обычных сайтов, а именно: *HTML*, *CSS* И *JavaScript.* Ключевым моментом интерактивных веб-сайтов является объектная модель документа (*DOM*). *DOM* – это специальный программный интерфейс (*API)* для документов, написанных на языке гипертекстовой разметки (*HTML)*. Концептуально *DOM* можно представить в виде дерева, состоящего из узлов и объектов, происходящих из корневого узла (самого документа) (рис. 1). Дочерние узлы разветвляются, чтобы сформировать его структуру. У них есть свойства и методы, которые позволяют манипулировать самими узлами.

Используя *DOM*, программисты могут создавать документы, перемещаться по их иерархии и изменять содержимое документа путем добавления, удаления или обновления узлов и содержимого. Эти модификации могут быть сделаны с использованием такого языка программирования, как *JavaScript. HTML* или язык гипертекстовой разметки позволяет создавать структуру страницы и используется для создания веб-страниц и веб-приложений [7].

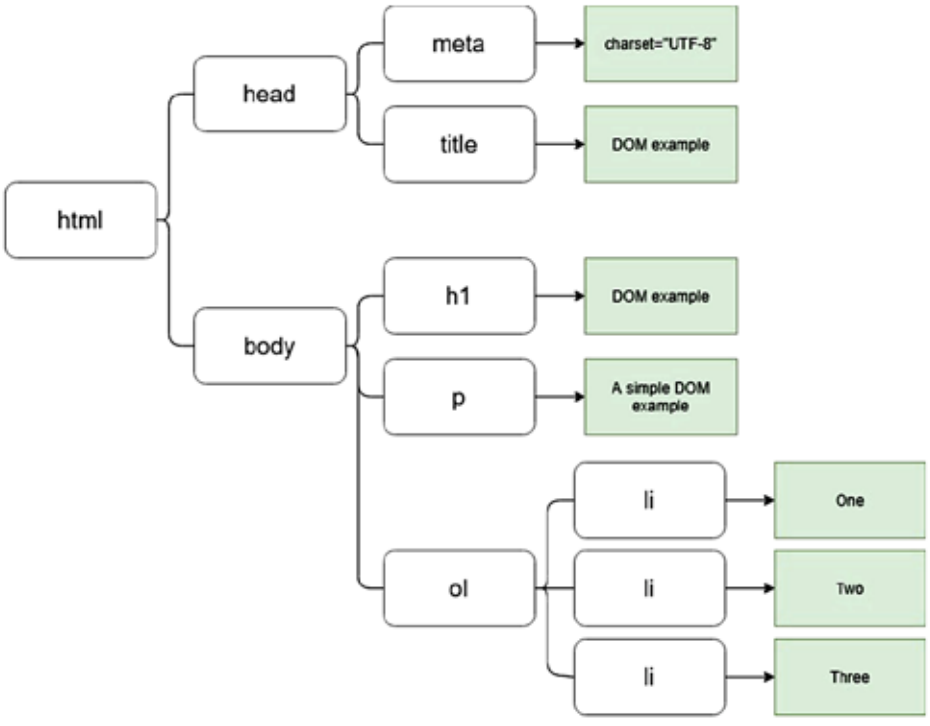

Рис. 1. Графическое представление DOM-узлов

CSS или каскадные таблицы стилей позволяют описывать внешний вид документа или веб-страницы, включая цвета, макет и шрифты. Это позволяет адаптировать веб-страницу к различным типам устройств, у которых большие экраны или маленькие экраны. *CSS* может использоваться с любым языком разметки на основе *XML* [5]. *JavaScript* – это язык программирования, который впервые появился в 1995 г. Вначале он использовался в разработке пользовательских интерфейсов для обеспечения интерактивности веб-сайтов, и, хотя это все еще наиболее распространенное применение *JavaScript*, в последние годы стало возможным использовать *JavaScript* на серверной части веб-приложений [8]. Этот тезис сосредоточен исключительно на клиентском *JavaScript*, который выполняется в браузере на клиенте. *JavaScript* соответствует международному стандарту *ECMA-262*, обычно называемому *ECMAScript*.

На сегодняшний день браузер *Google Chrome* является популярным среди остальных браузеров и включает в себя инструменты для замера производительности клиентских веб-сервисов [9].

Первый инструмент для оценки производительности из тех, которые встроены в браузер – это вкладка «Производительность», или «*Performance*», в такой вкладке можно выполнить для веб-приложения такие оценки производительности, как первая отрисовка, размер сборки, за какое время выполнится повторная перерисовка и т.д.

Оценку производительности осуществляют при помощи данного инструмента для того, чтобы:

− улучшить взаимодействие пользователя и приложения (пользовательский опыт);

− привлечь больше клиентов;

− увеличить пользу для владельцев продукта – количество пользователей или клиентов.

Вкладка «Производительность» в инструментах разработчика удобна, если разработчик или тестировщик сфокусированы на каком-то одном веб-приложении и тестируют его производительность, но с его помощью, например, будет сложно сравнить два или более идентичных веб-приложения, при этом не запуская каждый раз оценку вручную.

Следующим и самым распространенным из инструментов оценки производительности является *Lighthouse,* которое решает проблему поиска проблем производительности в сайтах или *Single Page Application* и является автоматизированным инструментом с открытым исходным кодом, позволяющим проанализировать метрики любого веб-портала.

Из основных метрик, на основе которых строятся результаты, можно выделить *Web Vitals,* метрики производительности, метрики доступности [10]. Более того, система способна в результате сформировать свод общих советов для улучшения производительности. Данный инструмент предоставляет большое количество возможностей и может использоваться не только как часть браузера, но так же и как программная библиотека (например *NodeJS* модуль). К сожалению, этот инструмент не способен запускать уникальные пользовательские тест-кейсы (только встроенные), которые могут быть предоставлены пользователями для тестирования веб-приложений, но недавно в блоге разработчиков *Google Chrome DevTools* был представлен новый инструмент *Recorder.* Он позволяет записывать, сохранять и воспроизводить пользовательские тестовые сценарии. Особенность этого инструмента состоит в том, что в его возможности входит все то, что может *Lighthouse*. Дополнительно *Recorder* дает пользователям редактировать тест-кейсы или замерять метрики производительности путем открытия готовой записи через вкладку «Производительность» в инструментах разработчика [11]. Данный инструмент является на данный момент тестовым и имеет большую вероятность обрести такую же популярность как *Lighthouse.*

Существуют такие популярные инструменты, как *JMeter* и *LoadRunner,* которые очень эффективны для выполнения тестирования производительности веб-приложений. Данные программы, так же как и *Lighthouse,* помогают измерить производительность для клиентской и серверной части веб-приложений. *JMeter* является программным обеспечением с открытым исходным кодом для измерения производительности. Изначально сфера использования ограничивалась тестированием веб-приложений, но с тех пор оно расширилось до других функций [12]. Данное приложение может использоваться для тестирования производительности как на статических, так и на динамических ресурсах, веб-динамических приложениях. Его можно использовать для имитации большой нагрузки на сервер, группу серверов, сеть или объект, чтобы проверить ее прочность или проанализировать общую производительность при различных типах нагрузки [4].

Из особенностей можно выделить следующее:

− поддержка широкого перечня приложений/протоколов, таких как: *HTTP/HTTPS, SOAP, FTP, TCP* и др.;

– поддержка мультиплатформенности благодаря использованию языка программирования высокого уровня *Java*;

– поддержка анализа загрузки сети;

– поддержка многопоточности, позволяющая выполнять параллельную обработку запросов или определенных операций;

– возможность использования сценариев тестирования;

– поддержка кэширования и офлайн-воспроизведения результатов тестирования.

Хочется отметить еще то, что *JMeter* не имеет возможности выполнять клиентский *JavaScript* для измерения производительности, но он может записывать выполнение *HTTP* запросов, которые будут отправляться клиентской частью веб-приложений, что позволяет выполнять тесты, которые измеряют время выполнения запроса и получение ответа для клиента.

LoadRunner – программное обеспечение, которое позволяет выполнять нагрузочное автоматизированное тестирование с использованием эмулирования одновременного большого количества пользователей, чтобы изучить поведение приложения при реальной нагрузке [4, 13]. К базовым особенностям *LoadRunner* относятся:

– поддержка множества приложений и протоколов: *HTTP/ HTTPS /HTML, SAP, FTP, RDP, .NET* и др. [4];

– возможность интеграции в следующие среды разработки *Microsoft Visual Studio* и *Eclipse* [4];

– наличие инструментов, позволяющих создавать уникальные тестовые сценарии;

– возможность контроля процесса выполнения сценария;

– наличие возможности интеграции с облаком.

JMeter и *LoadRunner* позволяют создавать внушительное количество запросов, используя несколько компьютеров, при управлении процессом с одного из них, и дополнительно поддерживать плагины сторонних разработчиков, что помогает пользователям добавить в уже имеющийся набор инструментов новые возможности. В стандартном наборе инструментов этих программ присутствует такой функционал, как логирование результатов тестов, и возможности отображения данных в виде таблиц и графиков.

Хочется отдельно выделить, что данное программное обеспечение дает возможность выполнять тестирование производительности на уровне протокола *HTTP*, но не на уровне взаимодействия с пользователем. Следствием этого является то, что не могут быть покрыты случаи, при которых должна учитываться скорость приложения при проведении замеров при уникальном пользовательском взаимодействии с самим приложением, потому что современные клиентские-веб приложения хоть и работают с сетевыми запросами, но в то же время необходимо учитывать и скорость работы пользовательского интерфейса приложения.

Существует другое решение, предложенное *Thiam Kian*. В его работе «*Web page performance analysis*» система замеряет не только время, затрачиваемое на отрисовку страницы в браузере, но и время, затрачиваемое на отправку данных веб-страницы с сервера на клиент. Данное решение решает проблему измерения производительности при взаимодействии клиента и сервера базируется на следующих элементах:

– прокси-сервер, настроенный и размещенный между клиентом и сервером для записи времени, прошедшего между запросом для получения *HTML*-страницы (и ее встроенных объектов);

– доступ к журналу сервера, где записывается время обработки каждого запроса;

– веб-страницы, оснащенные *JavaScript* для записи времени, затрачиваемого на рендеринг страницы в браузере.

Иллюстрация работы данного метода представлена на рис. 1.

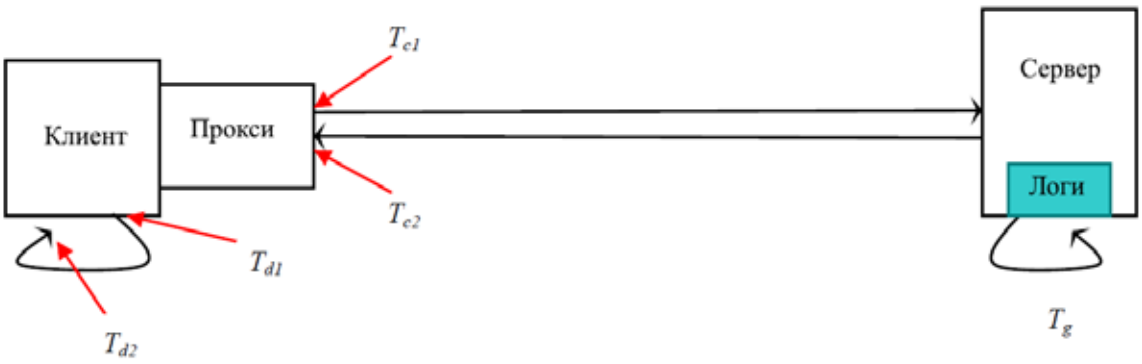

Рис. 2. Измерение и анализ времени отклика с помощью прокси-сервера

На рис. 2 изображены взаимодействие элементов системы измерения производительности клиентской и серверной части веб приложений с использованием прокси-сервера и логирования запросов на стороне сервера.

Для измерения метрик производительности были выведены следующие формулы:

$$\text{время ответа от сервера} = T_{d2} - T_{c1} \quad (1),$$

$$\text{время рендеринга страницы в браузере} = T_{d2} - T_{d1} \quad (2),$$

$$\text{время, затрачиваемое на передачу данных} = (T_{c2} - T_{c1}) - T_g \quad (3),$$

где T_{c1} – время, когда клиентский запрос был отправлен на сервер,

T_{c2} – время, когда прокси-сервер получил ответ от сервера,

T_{d1} – время, когда началась первая отрисовка контента в браузере,

T_{d2} – время, когда завершилась отрисовка контента в браузере,

T_g – время, за которое генерируется ответ на сервере.

Когда клиент запрашивает веб-страницу, запрос будет проанализирован и время будет записано прокси-сервером. Затем запрос пересылается на сервер. После обработки запроса сервером он возвращает клиенту соответствующий ответ. Информация об этой совпадающей паре, запрос-ответ, также будет записана в журнал сервера. Ответ сервера будет проанализирован, и время его получения – записан прокси-сервером до того, как браузер отобразит его. Все встроенные объекты пройдут через эти шаги [14].

Таким образом, время, когда каждый объект был запрошен и получен, вся информация будет записана прокси-сервером. На клиенте полученный ответ будет обработан с использованием *JavaScript* и *Web application programming interface*, поэтому, когда браузер запускает и завершает отображение веб-страницы, происходит расчет времени рендеринга для этой страницы.

Данный метод расчета производительности для веб-приложений имеет плюсы: время отклика для разных веб-страниц можно измерить и сравнить; время, необходимое для загрузки *HTML*-страницы и каждого из ее встроенных объектов, можно измерить отдельно; можно определить время, затрачиваемое сервером на обработку каждого *HTTP*-запроса.

Из минусов можно выделить то, что развертывание системы может быть большой проблемой, поскольку такая сложная задача должна включать в себя, например, разрешение администратору веб-сайта доступа к журналу *ISP* или *LAN* прокси-сервера. Поскольку журнал доступа к серверу необходим для поддержки измерений и анализа, клиенты, чьи имена или *IP*-адреса скрыты прокси-сервером, не могут быть идентифицированы. Если это произойдет, измерение и анализ станут более трудными и могут быть менее точными.

Заключение

Вопрос производительности клиентских-веб приложений был актуален как раньше, так и в настоящее время. Большие компании и отдельные разработчики предоставляют множество решений для измерений отдельных показателей веб-страницы или определенного функционала веб-приложений, однако проблема состоит в том, что большинство из имеющихся популярных решений имеют узкую сферу применения при измерении производительности. Они не могут решить все проблемы по измерению производительности для современных корпоративных клиентских веб-приложений.

Для этого нужно учитывать множество факторов при построении системы для измерения: доступность и архитектуру, возможность разворачивания, возможность использования другими разработчиками для интегрирования в процесс разработки, и дополнительно нужно учитывать уникальные измерения метрик под определенными нагрузками. К примеру, возможность тестировать производительность, используя уникальные пользовательские сценарии. Это даст более подробное представление о производительности, нежели система, которая делает общие тесты производительности и выводит общие рекомендации.

Список литературы

1. Cisco Annual Internet Report (2018–2023). [Электронный ресурс]. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (дата обращения: 28.11.2021).

2. СТО 610.12-1990 IEEE standard glossary of software engineering terminology.

3. Логинова Ф.С. Информационные технологии в социальной сфере: учебное пособие. СПб.: ИЭО СПбУТУиЭ, 2010. 250 с.

4. Темичев А.А., Файзрахманов Р.А. Аналитический обзор средств автоматизации тестирования производительности применительно к системам мониторинга // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. 2015. № 15. С. 117–133.

5. What Are Core Web Vitals? 4 Standards That Matter. [Электронный ресурс]. URL: https://www.promisemedia.com/content-development/core-web-vitals (дата обращения: 01.12.2021).

6. Timing for bringing page experience to Google Search. [Электронный ресурс]. URL: https://developers.google.com/search/blog/2020/11/timing-for-page-experience (дата обращения: 03.12.2021).

7. HTML & CSS – W3C. [Электронный ресурс]. URL: https://www.w3.org/standards/webdesign/htmlcss (дата обращения: 03.12.2021).

8. Хавербеке М. Выразительный JavaScript. Современное веб-программирование. М.: Питер, 2019. 320 с.

9. Названы самые популярные браузеры: Софт: Наука и техника: Lenta.ru. [Электронный ресурс]. URL: https://lenta.ru/news/2021/04/06/browser/ (дата обращения: 03.12.2021).

10. Lighthouse | Tools for Web Developers | Google Developers. [Электронный ресурс]. M URL: https://developers.google.com/web/tools/lighthouse (дата обращения: 09.10.2021).

11. Record, replay and measure user flows – Chrome Developers. [Электронный ресурс]. URL: https://developer.chrome.com/docs/devtools/recorder/ (дата обращения: 09.10.2021).

12. Apache JMeter. [Электронный ресурс]. URL: https://jmeter.apache.org/ (дата обращения: 09.11.2021).

13. LoadRunner: Application Load Testing Tools | Micro Focus. [Электронный ресурс]. URL: https://www.microfocus.com/en-us/products/loadrunner-professional/overview (дата обращения: 09.10.2021).

14. Chiew Thiam Kian. Web page performance analysis. University of Glasgow, 2009. P. 75-85.