На правах рукописи

МАКСИМОВ ЯРОСЛАВ АЛЕКСАНДРОВИЧ

МЕТОДЫ, МОДЕЛИ И МЕТРИКИ СБОРА ДАННЫХ О ПРОИЗВОДИТЕЛЬНОСТИ КЛИЕНТ-СЕРВЕРНЫХ ВЕБ-ПРИЛОЖЕНИЙ

Специальность 2.3.8. – Информатика и информационные процессы
(технические науки)

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

ПЕНЗА – 2025

Работа выполнена на кафедре «Программирование» в Федеральном государственном бюджетном образовательном учреждении высшего образования «Пензенский государственный технологический университет».

Научный руководитель – **Мартышкин Алексей Иванович** кандидат технических наук, доцент, заведующий кафедрой «Программирование» ФГБОУ ВО «Пензенский государственный технологический университет», г. Пенза

Официальные оппоненты: **Тарасов Вениамин Николаевич**
доктор технических наук, профессор, заслуженный работник высшей школы РФ, профессор кафедры информатики и робототехнических систем федерального государственного бюджетного образовательного учреждения высшего образования «Поволжский государственный университет телекоммуникаций и информатики», г. Самара

Ямашкин Станислав Анатольевич
кандидат технических наук, доцент, доцент кафедры автоматизированных систем обработки информации и управления федерального государственного бюджетного образовательного учреждения высшего образования «Национальный исследовательский Мордовский государственный университет им. Н.П. Огарёва», г. Саранск

Ведущая организация – Федеральное государственное бюджетное образовательное учреждение высшего образования «Юго-Западный государственный университет», г. Курск

Защита диссертации состоится «22» декабря 2025 года в 15:00 на заседании объединенного диссертационного совета 99.2.113.02 на базе ФГБОУ ВО «Рязанский государственный радиотехнический университет имени В.Ф. Уткина», ФГБОУ ВО «Пензенский государственный технологический университет» по адресу: 390005, г. Рязань, ул. Гагарина, д. 59/1.

С диссертацией можно ознакомиться в научной библиотеке ФГБОУ ВО «Рязанский государственный радиотехнический университет им. В.Ф. Уткина, на сайте http://rsreu.ru/ и в научной библиотеке ФГБОУ ВО «Пензенский государственный технологический университет».

Автореферат разослан «___» _____2025 г.

Ученый секретарь
диссертационного совета
доктор технических наук, доцент А.Н. Колесенков

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность. Для современного мира характерен стремительный информационно-технологический прогресс коммерческих организаций с расширением их поля деятельности за счет интернет-ресурсов, при этом как государственные сервисы, так и торговые и банковские интернет-услуги представлены в основном посредством веб-приложений, большая часть из которых реализована на базе архитектуры «клиент-сервер», доступной для пользователей через всемирную сеть.

Интернет, который сложился к сегодняшнему времени, развивался на протяжении многих лет благодаря созданию новых подходов и технологий разработки веб-приложений. На ранних этапах развития всемирной паутины основным методом генерации веб-страниц оставался серверный рендеринг (*Server-Side Rendering, SSR*), когда формирование итоговой *HTML*-страницы выполнялось непосредственно на стороне сервера. При каждом запросе клиента сервер выполнял обработку данных, сборку компонентов приложения и генерацию *HTML*-страницы, содержащей как статический контент, так и динамические элементы. Сформированный документ передавался в браузер пользователя, где мог быть отображен без дополнительных операций по изменению контента внутри страницы.

Переломным моментом в эволюции веб-приложений стало появление концепции *Web* 2.0, ознаменовавшей собой переход от статических страниц к интерактивным веб-приложениям. Именно в этот период веб-сайты стали предоставлять пользователям возможность динамического изменения внутренней структуры веб-страницы в режиме реального времени. Существенную роль в этом сыграло распространение асинхронного *JavaScript* (*AJAX*) – новой парадигмы обработки клиент-серверных запросов, основанной на асинхронной модели взаимодействия. Реализации *AJAX* быстро получили распространение в популярных *JavaScript*-библиотеках, таких как *jQuery* и *BackboneJS*, которые предоставляли разработчикам удобные абстракции для работы с *DOM* и для организации интерактивных интерфейсов. Данный прорыв в сфере *веб-технологий* способствовал формированию устойчивых архитектурных шаблонов пользовательских интерфейсов, что, в свою очередь, стимулировало переход от многостраничных веб-приложений к архитектуре одностраничных приложений (*SPA, Single Page Application*). Основной идеей этого перехода стало смещение части бизнес-логики на сторону клиента, что позволило снизить нагрузку на веб-серверы и повысить их производительность при обработке большого количества запросов.

Однако с ростом сложности клиентской логики возникла необходимость в более масштабируемых и контролируемых архитектурных решениях. Появление компонентных подходов к построению интерфейсов, таких как архитектура *Flux*, представленная инженером-программистом *J. Chen* в 2014 г., ознаменовало отказ от традиционной модели *Model-View-Controller* (*MVC*) в пользу однонаправленного потока данных и централизованного управления состоянием приложения. Использование данного подхода обеспечивало предсказуемость поведения клиентских приложений. Спустя десятилетие в современных веб-разработках вновь наблюдается возрождение интереса к серверному рендерингу, который позволяет быстрее загружать страницы, причем сгенерированные страницы индексируются поисковыми системами. Современные реализации *SSR*, такие как фреймворк *Next.js*, сочетают преимущества серверного и клиентского рендеринга, обеспечивая улучшенную *SEO*-оптимизацию и предоставляя интерфейс для создания веб-страниц. Популярность метода *SSR* в современных условиях объясняется также тем, что он решает задачи первоначальной загрузки, доступности и индексации поисковыми системами, что делает его важным компонентом гибридных архитектур веб-приложений.

Серверный рендеринг ныне рассматривается как один из наиболее эффективных подходов для повышения масштабируемости веб-приложений и обеспечения консистентности передаваемых данных, поскольку логика формирования страницы и агрегации данных выполняется централизованно на стороне приложения. Пользователь получает уже полностью подготовленный *HTML*-документ, снижая риск ошибок, связанных

с несовместимостью клиентских сред или ограничениями вычислительных ресурсов конечного устройства. Однако вместе с этими преимуществами возникли и новые проблемы, связанные с усложнением логики генерации страниц, увеличением числа обращений к базе данных и ростом объема передаваемого контента. В результате время отклика стало зависеть не только от характеристик сети, но и от структуры веб‑страницы.

В процессе работы веб-приложения формируются данные о производительности, которые содержат следующие метрики: на стороне клиента – время отклика, время рендеринга, значение сдвига элементов пользовательского интерфейса, а на стороне сервера – количество запросов в минуту, время задержки, загрузка процессора. Компании используют указанные метрики для оценки корректности работы и повышения качества предоставляемых сервисов. Практика показывает, что своевременный анализ подобных данных помогает сократить число отказов и повысить устойчивость приложений за счет внедрения проверенных инженерных решений.

Исследования в области производительности клиент-серверных веб-приложений опираются на ряд фундаментальных работ, авторами которых являются *Raj Jain* и *David J. Lilja*, которые заложили теоретические основы для этой области. Работы *Jakob Nielsen* и *Feng F.-H. Nah* связали метрики отклика с влиянием на пользователей. Из современных исследователей в этой области можно выделить *Patrick Meenan, Tammy Everts,* А.Е. Кучерявого, А.М. Сухова, Л. Черкасова, *Albert Greenberg, Kimura Hideaki*, И. Григорика.

Исследования, направленные на анализ производительности веб-приложений, играют ключевую роль в формировании методологических и инструментальных основ оценки эффективности функционирования клиент-серверных систем.

Полученные в ходе исследования результаты позволяют улучшить метрики веб‑приложений, повысить надежность и предсказуемость работы клиент-серверных систем, а также усовершенствовать пользовательское взаимодействие за счет сокращения времени отклика и повышения стабильности загрузки контента. Улучшение данных аспектов веб-приложений способствует не только удержанию сложившейся аудитории благодаря улучшению качества обслуживания, но и расширению пользовательской базы, поскольку качество программного обеспечения и его надежность являются определяющими факторами конкурентоспособности современных веб‑решений.

Объектом исследования является быстродействие веб-приложений на клиентской стороне, рассматриваемое в контексте анализа производительности в условиях взаимодействия с сервером.

В качестве **предмета исследования** рассматриваются совокупность методов и алгоритмов, применяемых для сбора данных метрик, характеризующих быстродействие клиентской части веб-приложений в клиент-серверной архитектуре.

Цель работы заключается в снижении времени отклика клиент-серверных веб-приложений на основе разработанных моделей и методик сбора данных путем создания технических решений, обеспечивающих получение и интерпретацию информации, полученной в виде метрик.

Для достижения поставленной цели решен ряд **задач**.

1. Проведен обзор существующих решений, методов анализа и оценки производительности клиент-серверных веб-приложений.

2. Разработана математическая модель генерации веб-страницы, которая учитывает влияние таких метрик как количество строк кода, сложность программных конструкций и состояние приложения (количество подключений к базе данных, количество переменных *cookie*) на время отклика для сгенерированной веб-страницы.

3. Разработана математическая модель контента веб-страницы, описывающая влияние таких метрик контента как общий размер *HTML*-документа, количество дополнительных объектов, размер и количество *JavaScript* и *CSS*-ресурсов на время отклика.

4. Разработан метод численной оценки производительности клиент-серверных веб-приложений, учитывающий свойства клиентской части взаимодействия.

5. Разработана методика анализа производительности веб‑страниц клиент‑серверных приложений, основанная на интеграции метода сбора данных о производительности в программном комплексе, математических моделей и алгоритмов.

6. Выполнена верификация разработанного метода, моделей, методики и предложены пути их применения в реальных условиях использования.

В процессе выполнения исследований и разработки алгоритмического и программного обеспечения, направленных на решение задач, поставленных в данной диссертационной работе, использованы методы вычислительной математики, линейной алгебры, математического моделирования.

Научная новизна работы заключается в следующем.

1. Создан метод сбора данных о производительности клиент-серверных приложений с использованием программного комплекса, состоящего из отдельных масштабируемых компонентов микросервисной архитектуры, отличающийся встроенной в клиентские веб-страницы библиотекой-агентом, что позволяет снизить накладные расходы на выполнение сбора данных.

2. Разработана математическая модель генерации веб-страницы, отражающая влияние на время отклика структурных метрик веб-страниц, число строк кода, сложность программных конструкций и состояние приложения, отличающаяся формализацией синтаксических конструкций, что дает возможность количественно оценить сложность формирования страницы на сервере еще до передачи клиенту.

3. Разработана математическая модель контента веб-страницы, отличающаяся представлением контент веб-страницы активным фактором, влияющим на работу клиентской части приложения одновременно с метриками контента, что позволяет осуществить количественную оценку производительности.

4. Предложена новая методика и алгоритм оценки свойств веб-страницы по внутренним метрикам страницы, которые в отличие от аналогов, учитывающих лишь функции сетевых и серверных факторов, учитывают синтаксическую структуру кода, динамическую логику генерации и содержимое страницы, что дает возможность получения единого и полного критерия оценки.

Соответствие паспорту научной специальности. Область научного исследования, регламентированная в паспорте специальности 2.3.8. – Информатика и информационные процессы, включает следующие направления:

– разработка компьютерных методов и моделей описания, оценки и оптимизации информационных процессов и ресурсов, а также средств анализа и выявления закономерностей на основе обмена информацией пользователями и возможностей используемого программно-аппаратного обеспечения (п. 1);

– разработка архитектур программно-аппаратных комплексов поддержки цифровых технологий сбора, хранения и передачи информации в инфокоммуникационных системах, в том числе, с использованием «облачных» интернет-технологий и оценок их эффективности (п. 9).

Теоретическая значимость. Совершенствование методов, моделей и метрик для сбора и анализа данных о производительности клиент-серверных веб-приложений.

Практическая ценность. Применение предложенных в диссертации методов, моделей и алгоритмов помогает выявить факторы, негативно влияющие на время отклика клиент-серверных веб-приложений. Используя предложенные методы, модели и метрики, разработчик сократит затраты времени на улучшение работы клиент-серверных приложений, а также их поддержку.

Реализация и внедрение результатов работы. Разработанные методы и алгоритмы внедрены в учебный процесс на кафедре «Программирование» ФГБОУ ВО ПензГТУ и используются при подготовке студентов по направлениям бакалавриата 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» в рамках

дисциплин «Основы *DevOps* и *DataOps*», «Языки интернет программирования», «Сети и телекоммуникации», «Администрирование информационно-коммуникационных систем», «Компьютерные сети», а также при подготовке студентов по направлению магистратуры 09.04.04 «Программная инженерия» в рамках дисциплин «Распределенные системы обработки информации», «Протоколы вычислительных сетей». Отдельные программно-технические решения, созданные в ходе диссертационного исследования, использованы в АО «НПП «Рубин» (г. Пенза) при модификации специального программного обеспечения автоматизированной системы управления материально-технического обеспечения, в АО «Радиозавод» (г. Пенза) при разработке и поддержке программного обеспечения производственного процесса.

Достоверность результатов работы подтверждается их непротиворечивостью результатам других исследователей, внедрением в практическую и научно-исследовательскую деятельность ряда организаций, а также апробацией и одобрением на всероссийских и международных научных конференциях.

На защиту выносятся:

1. Метод сбора данных о производительности клиент-серверных веб-приложений который реализуется с помощью программного комплекса и позволяющий фиксировать ключевые метрики производительности в реальном времени, организовывать их централизованный сбор и последующую агрегацию без необходимости модификации серверной логики анализируемых приложений. Программный комплекс обеспечивает снижение времени отклика веб-страниц в среднем на 30% по сравнению с традиционным методом, основанным на прокси-сервере, а также его применение позволило снизить время на диагностику проблем на 13,7% в АО «Радиозавод» (г. Пенза).

2. Математическая модель генерации веб-страницы, позволяющая количественно оценивать вычислительную сложность формирования страницы на стороне сервера с помощью метрик числа строк кода, сложности программных конструкций и состояния приложения, а также выявлять потенциально узкие места при ее создании и определять на этапе проектирования влияние серверной нагрузки на время отклика.

3. Математическая модель контента, количественно описывающая влияние метрик, связанных с контентом страницы: общий размер *HTML*-документа, количество дополнительных объектов, размер и количество *JavaScript* и *CSS*-ресурсов, позволяющая оценивать влияние наполнения веб-страницы на ее производительность.

4. Методика анализа производительности веб-страниц и алгоритм на ее основе, позволяющие реализовать непрерывный контроль производительности веб-приложений, автоматизировать сбор и интерпретацию данных в процессе их использования.

Апробация работы. Основные результаты, полученные в рамках диссертационного исследования, были опубликованы в научных журналах и апробированы на международных и всероссийских научных конференциях: V Всероссийская научно-практическая конференция «Актуальные вопросы современной науки: теория и практика научных исследований» (Пенза, 2021); Международная научно-практическая конференция разработчиков веб-технологий «*Web guild conference*» (Цюрих, 2021); Международная научно-практическая конференция «Современные информационные технологии» (Пенза, 2020, 2023, 2024, 2025); Международная научно-практическая конференция «Проминжиниринг» (*ICIE*) (Сочи, 2025); XXIV Всероссийская научно-практическая конференция «Молодые учёные России» (Пенза, 2025).

По результатам диссертационного исследования опубликовано 15 научных работ, в том числе 5 статей в журналах, рекомендованных ВАК Минобрнауки России, 1 статья, индексируемая в международной базе данных *Scopus*, получено 2 свидетельства о государственной регистрации программ для ЭВМ.

Личный вклад автора. Все представленные в работе результаты исследования являются оригинальными и были получены автором самостоятельно. Данные,

заимствованные у других авторов, сопровождаются ссылками на соответствующие опубликованные источники.

Объем и структура диссертации. Работа состоит из введения, четырех глав, заключения, списка литературы, который включает 115 наименований, и 2 приложений. Общий объем диссертации составляет 143 страницы. Диссертация содержит 23 таблицы и 23 рисунка.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность темы исследования, описываются предпосылки создания систем для сбора данных производительности и их важность для современной сферы веб-решений, а также формулируется цель работы и задачи, решение которых способствует достижению поставленной цели, отражены научная новизна и практическая значимость, сформулированы выносимые на защиту основные положения.

В первой главе представлен обзор истории развития веб-технологий с учетом эволюции клиент-серверных приложений в рамках веба. В работе рассматривается, каким образом поэтапное усложнение клиентской и серверной логики и архитектуры привело к возникновению проблем, связанных с производительностью. Отдельно в рамках исследования были проанализированы проблемные места, существующие как на стороне клиента, так и на стороне сервера.

Компьютерные системы оказывают значительное влияние на многие сферы деятельности человека, автоматизируя рутинные задачи, предоставляя информацию и доступ к цифровым сервисам для производств, бизнеса и государственных услуг. Эффективность таких систем, оцениваемая на основе объема выполненной работы за единицу времени при заданных ресурсных ограничениях, является важным параметром. Его необходимо учитывать на всех этапах жизненного цикла программного обеспечения от проектирования до поддержки. Веб-технологии, являясь одной из наиболее распространенных форм компьютерных систем, требуют комплексного анализа их производительности, включающего такие параметры, как скорость обработки запросов, использование вычислительных ресурсов и время отклика. Традиционно оценка производительности веб-систем осуществляется с позиций серверной инфраструктуры и характеристик сетевого взаимодействия. Однако в данном исследовании предлагается усовершенствованный метод, направленный на анализ производительности клиент-серверных веб-приложений, включающий автоматизированный сбор метрик, моделирование факторов, влияющих на время отклика, и интерпретацию результатов с целью выявления узких мест как на стороне клиента, так и на стороне сервера. В качестве ключевого показателя эффективности рассматривается метрика времени отклика, определяемая как метрика, основанная на внутренней структуре страницы в рамках клиент-серверного приложения. Для проведения исследования разработано специализированное программное обеспечение для сбора данных, а также предложены модели и методика для оценки времени отклика.

В рамках первой главы проведено исследование отечественных и иностранных патентов, которые содержательно схожи с данным исследованием. Первый патент RU2669172C2 за авторством Канцева И. В. и Егорова Д. С. предлагает систему и метод мониторинга согласованности веб-сайтов, ориентированные на автоматизированное выявление ошибок в выводе данных и других эксплуатационных проблем. Второй патент, RU2680746C2 за авторством Цзыню Ч. из Китая, описывает метод и устройство для формирования моделей качества веб-страниц, направленные на повышение точности их оценки. Известна также разработка, направленная на повышение производительности веб-страниц через управление задержкой тегов, по патенту RU2763000C2. Данное решение базируется на концепции контроля состояний тегов и их асинхронного мониторинга. Наряду с запатентованными решениями, в научно-технической литературе представлены методы, с которыми сравниваются представленные в данной работе результаты для оценки их достоинств и недостатков.

Метод сбора данных о производительности клиент-серверных веб-приложений на основе прокси-сервера реализуется посредством перехвата сетевого трафика между двумя

узлами данной архитектуры. В рамках метода вводится промежуточный компонент, который называется «прямой прокси», через который проходят *HTTP(S)*-запросы (*WebScarab, Muffin*). Это позволяет зафиксировать каждый запрос от клиента и соответствующий ему ответ от сервера, определить время передачи, объем данных, а также последовательность и длительность операций на сетевом уровне. Метод не требует модификации ни клиентской, ни серверной части приложения, что делает его универсальным по отношению к объекту исследования.

Следующий метод основан на использовании вложенных *HTML*-страниц или так называемых фреймов, обозначаемых тегом *<iframe>*. Он позволяет собирать данные о времени отклика и времени рендеринга через загрузку тестируемой страницы во фрейм, встроенный в управляющий контейнер. *JavaScript*-код во внешнем фрейме фиксирует ключевые временные события, позволяя оценить продолжительность загрузки и отображения целевой страницы. Метод не требует изменения структуры анализируемого ресурса и может применяться при соблюдении политики одного источника.

Среди прочих энтерпрайз решений, которые используются на клиенте, можно выделить *Web Vitals*, которое предоставляет разработчику набор метрик, однако не раскрывает зависимости типа причина-следствие. *Lighthouse* ориентирован преимущественно на статический анализ и рекомендации по оптимизации, но не предназначен для анализа производительности в условиях реальной эксплуатации.

Цели вышеописанных патентов и методов совпадают с разработкой, представленной в настоящем исследовании по той причине, что во всех случаях рассматривается задача анализа и мониторинга веб-приложений. Однако предлагаемые в патентах решения носят фрагментарный характер и направлены на устранение отдельных проблем. В отличие от них, в диссертации разработан системный подход, объединяющий разрозненные части в единую методику.

В главе рассматриваются основные механизмы работы браузера, обеспечивающие визуализацию веб-страниц, а также основные программные интерфейсы, используемые в процессе их загрузки и отображения. Особое внимание уделено интерфейсу *PerformanceTiming* (рисунок 1), который имеет важную роль при анализе и работе с клиентской частью веб-приложений.

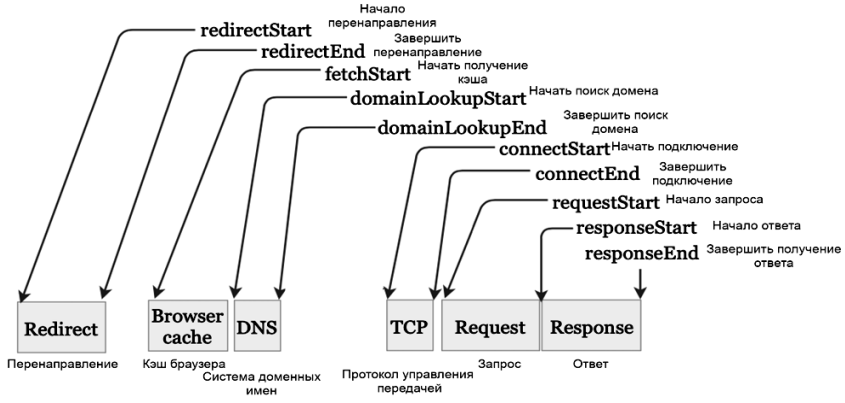

Рисунок 1 – Интерфейсы браузера *PerformanceTiming* и *PerformanceNavigation*

Интерфейс *PerformanceTiming* предоставляет разработчику доступ к временным метрикам, которые связаны с основными этапами загрузки веб-страницы. Полученные данные позволяют проанализировать время загрузки приложения и выявлять узкие места, влияющие на производительность. В рамках работы сбор данных производится как на клиентской, так и на серверной частях веб-приложений. Клиентские веб-приложения могут опираться на программные интерфейсы браузера, которые позволяют извлечь необходимую информацию о взаимодействии пользователя с системой и на ее основе создать метки, отражающие метрики пользовательского поведения и характеристики отзывчивости приложения.

Производительность серверных приложений оказывает существенное влияние на общее восприятие пользователем и эффективность реализации бизнес-целей (конверсия и удержание клиентов). Одним из основных показателей является то, сколько запросов в секунду обрабатывает то или иное приложение под нагрузкой, время отклика, включающее время обработки запроса на сервере и передачу данных пользователю. Такие данные могут быть получены из логов сервера (рисунок 2).

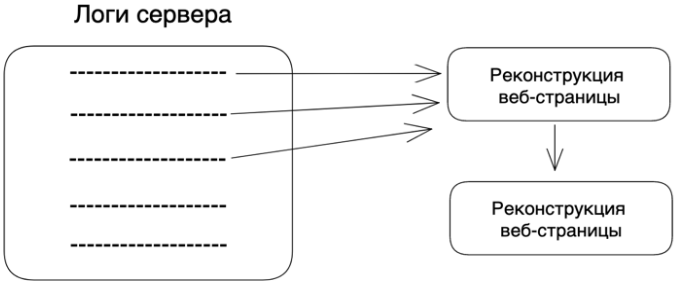

Рисунок 2 – Шаги для анализа логов сервера для оценки времени отклика

Их отличительной особенностью является то, что они обладают разной структурой, что усложняет их обработку. Для решения этих проблем в рамках данной работы было решено использовать расширенный формат логов от *W3C*, которые позволяют стандартизировать подход к записи в такой файл и получению информации из него.

Знание тонкостей работы клиент-серверных приложений, а также каждого из компонентов, в частности, позволяет определить их узкие места производительности для проведения более глубокого анализа для создания решений проблем производительности и их безопасности.

Во второй главе на основе результатов проведенного анализа в рамках предыдущей главы приводятся решения в рамках сбора данных производительности. В рамках работы рассматривается клиент-серверное взаимодействие, при котором веб-страницы генерируются на сервере, отдаются клиенту и отображаются посредством браузера. В таком взаимодействии участвуют три компонента: клиент, сервер и сеть (рисунок 3). Каждый из них имеет определенный круг задач, в рамках которых все выполненные операции занимают время, что в свою очередь отражается на общем времени отклика.

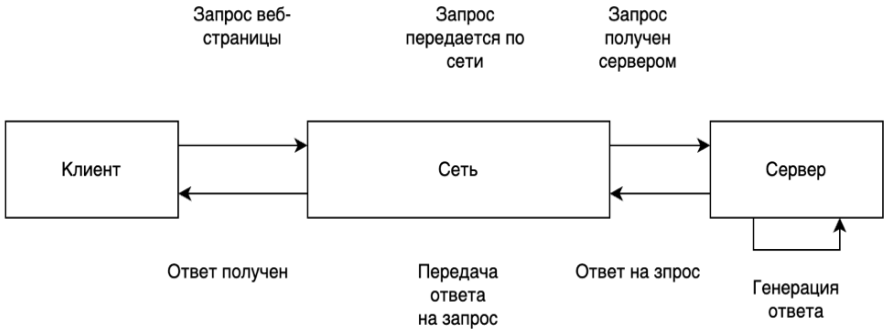

Рисунок 3 – Компоненты, участвующие в клиент-серверном взаимодействии

Метод сбора данных производительности клиент-серверных веб-приложений с использованием программного комплекса позволяет собирать ключевые метрики веб-приложений, имеющих клиент-серверную архитектуру непосредственно в среде его выполнения, обеспечивая точное профилирование. Метод реализуется с помощью встроенной библиотеки-агента на *JavaScript*, которая выполняет сбор данных метрик, связанных с рендерингом, временем отклика и загрузкой ресурсов. Благодаря интеграции в клиентский код разработчик получает доступ к деталям внутреннего исполнения, что делает решение гибким и подходящим для таких архитектур. В результате полученное время отклика приближено к реальному пользовательскому опыту. При этом необходимо учесть, чтобы каждый запрос выполнялся через страницу, включающую библиотеку сбора

данных, в противном случае метрики не будут зафиксированы системой. Данные о времени отклика, передачи и рендеринга страниц, а также параметры загрузки ресурсов фиксирует библиотека-агент. На серверной стороне осуществляется сбор логов и технических характеристик, отражающих процессы генерации страниц, обращения к базе данных и сетевые операции. Программный комплекс обеспечивает синхронность сбора данных, что позволяет соотнести метрики, формализующие генерацию и контента веб-страниц, с метриками, измеряемыми в ходе выполнения тестовых сценариев.

Взаимодействие клиента и сервера для разработанного метода можно выразить следующим образом:

$$T_{OTVSERV} = T_{R2} - T_{K1}, \quad (1)$$

$$T_{RB} = T_{R2} - T_{R1}, \quad (2)$$

$$T_{PD} = \left(T_{K2} - T_{K1}\right) - T_G. \quad (3)$$

где $T_{OTVSERV}$ – время, когда произошел отклик от сервера,

T_{RB} – время, за которое веб-страница была сформирована,

T_{PD} – время передачи данных,

T_{K1} – время отправки запроса на сервер,

T_{K2} – время получения клиентом ответа от сервера,

T_{R1} – время начала отображения содержимого страницы в браузере,

T_{R2} – время завершения отображения контента в браузере,

T_G – время генерации ответа на сервере.

Данный метод отличается низкими затратами на внедрение и позволяет с высокой точностью оценивать совокупное время отклика, воспринимаемое конечным пользователем. Несмотря на то, что указанный метод не позволяет в полной мере проанализировать вклад таких отдельных компонентов страницы как *HTML*-разметка или дополнительные объекты в общее время отклика, он не требует изменений на стороне сервера или модификации содержимого веб-страниц.

В рамках работы был проведен анализ разработанного метода и сравнение с классическим методом на основе прокси-сервера и методом, основанном на фреймах. Было создано тестовое веб-приложение для отображения библиотек *JavaScript*, а также сценарии для его тестирования. Более того в рамках экспериментальной части исследования были проведены тесты, направленные на исследование разработанного метода и оценку влияния различных факторов на производительность веб‑страниц:

1. При первом сценарии происходит исследование поведение клиента при повторном обращении к уже ранее загруженным ресурсам веб-страницы. Задаются различные стратегии кэширования: отсутствие кэша, кэш с валидацией и кэш без валидации. Целью теста является определение того, насколько предлагаемые методы сбора данных производительности способны учитывать влияние кэширования на передачу ресурсов и общее время отклика. Данный аспект важен для корректной интерпретации метрик при анализе поведения страниц.

2. В следующем тестовом сценарии проводится сравнение статических и динамических веб-страниц, имеющих схожую структуру и контент. Однако, они различаются по способу создания. Одна из них представлена как предварительно сгенерированная *HTML*-страница, другая генерируется при каждом запросе. Сценарий позволяет подтвердить, что динамическая генерация контента вносит задержку, и предлагаемые методы способны зафиксировать данный эффект.

3. В рамках следующего сценария проведено исследование взаимосвязей между метриками веб-страниц: временем генерации на сервере, временем передачи данных и временем отображения на клиенте. Для этого были собраны эмпирические данные с

использованием описанных методов, после чего выполнен корреляционный анализ. Полученные выводы легли в основу модели формирования общего времени отклика.

Результаты тестов показали, что использование разработанного метода сбора данных на основе программного комплекса обеспечивает снижение времени отклика веб-страниц в среднем на 30 % по сравнению с традиционным методом, основанным на прокси-сервере. Такой эффект достигнут за счет исключения промежуточного сетевого слоя, отсутствия дополнительной маршрутизации *HTTP*-трафика и минимизации вычислительных расходов. Проведенные эксперименты подтвердили, что программный комплекс не вносит дополнительных временных задержек в ходе сбора данных.

В рамках работы были предложены математические модели генерации и контента. Первая модель строится на основе метрик, учитывающих количество строк кода, сложность используемых программных конструкций, характеристик запросов к базе данных Вторая учитывает метрики контента (размер, число встроенных объектов, использование *CSS/JS*).

Термин «сложность» отражает затраты вычислительных на формирование и отображение веб-страницы или ее контента в контексте моделей. Однако для них накладываются ограничения применимости. Они связаны с классической архитектурой клиент-серверных веб-приложений, в которых *HTML*-код формируется на стороне сервера. Модели отражают взаимосвязь метрик кода и контента с временем отклика в условиях контролируемого эксперимента и стабильной сетевой среды. При этом предполагается постоянство аппаратной конфигурации сервера и клиента и исключается влияние механизмов кэширования и сетей доставки контента. Более того они учитывают количество и типы запросов к базе данных, однако не описывают такие внутренние механизмы СУБД как индексация, параллельное выполнение.

Для динамических страниц сложность генерации веб-страницы (C_1) выражается как функция от строк кода ($linecode$), количество конструкций языка программирования ($skprog$) и время доступа к базе данных ($dbacc$). Общее выражение может быть представлено как сумма частных функций, отражающих вклад отдельных факторов:

$$C_1 = g_{line}\left(linecode\right) + g_{prog}\left(skprog\right) + g_{db}\left(dbacc\right), \quad (4)$$

где $linecode$ – количество строк кода;

$skprog$ – сложность программных конструкций;

$dbacc$ – влияние состояния базы данных;

$g_{line}\left(linecode\right)$ – вклад количества строк кода приложения;

$g_{prog}\left(skprog\right)$ – вклад программных конструкций;

$g_{db}\left(dbacc\right)$ – вклад состояния базы данных.

Вкладом называется добавка от конкретной метрики в суммарную сложность.

Вклад количества выполняемых строк кода для генерации страницы выражается следующим образом:

$$g_{line}\left(linecode\right) = x \sum_{k=1}^{linecode} weight_{1,k}, \quad (5)$$

где $g_{line}\left(linecode\right)$ – вклад количества строк кода приложения в сложность веб-страницы,

x – коэффициент, увеличивающий вес строки кода для динамических страниц по сравнению со статическими и имеет ограничение $x \geq 1$,

$linecode$ – количество строк кода,

$weight_{1,k}$ – вес выполнения, присваиваемый каждой строке кода.

Страницу, содержащую j-программных конструкций, которые представляют собой циклы и функции, можно представить следующим образом:

$$\sum_{i=1}^{j} f\left(n_i\right) \cdot linecode_i \cdot weight_{2,i} \cdot d_i, \quad (6)$$

где $1 \le i \le j$;

$weight_{2,i}$ – вес выполнения, присвоенный строке кода конструкции;

$f(n_i)$ – сложность конструкции в нотации «Большое О»;

$linecode_i$ – количество строк кода;

d_i – глубина вложенности программной конструкции.

Во время работы приложения, когда сервер генерирует страницу, может быть установлено k_{soed} к j базам данных. Каждая такая база данных с размером sz_{db_i}, где $1 \le i \le j$, и для каждого db_i существует n_i запросов, сложность генерации которых зависит от количества баз данных j, определяется следующим образом:

$$g_{db}(dbacc) = weight_3 \cdot k_{soed} + \sum_{i=1}^{j}\left(n_i \cdot T_{sz_{db_i}}\right) + vl, \quad (7)$$

где $weight_3$ – вес для времени, необходимого для установления соединения с базой данных;

k_{soed} – количество установленных соединений к базе данных;

$T_{sz_{db_i}}$ – отражает время обработки запроса к базе данных с размером sz_{db_i};

vl – показатель влияния сессионных переменных *cookie*.

Серверная часть веб-приложений всегда имеет базу данных и часто общается с ней. В связи с этим фактом эта часть является основополагающей при генерации веб-страниц и приложений в целом. Их состояние поддерживается с помощью базы данных на сервере или с помощью *cookies* на клиентской машине. В контексте базы данных рассматривается метрика, обозначаемая как col_{db}. Она отражает количество обращений к базе данных. Размер базы данных обозначается как sz_{db}, а k_{soed} является количеством установленных соединений. Сложность генерации веб-страницы, связанная с доступом к базе данных, а также с обработкой данных сессий и *cookies*, может быть выражена следующим образом:

$$g_{db}(dbacc) = g_{conn}(k_{soed}) + g_{req}(ql, sz_{db}) + vl, \quad (8)$$

где $g_{conn}(k_{soed})$ – функция от времени установления соединений с базой данных;

$g_{req}(ql, sz_{db})$ – вклад количества запросов (ql) и размера базы данных (sz_{db});

vl – показатель влияния сессионных переменных *cookie*.

В рамках данной модели не учитываются скорость физического носителя, количество элементов и схема базы данных. Таким образом, можно количественно оценить три параметра, связанных со сложностью генерации динамической веб-страницы, которыми являются $linecode$, $skprog$, $dbacc$.

Сложность контента веб-страницы (C_2) можно выразить как сумму всех весов сложности для всех дополнительных объектов как $C_2 = \sum_{i=1}^{j} w_{elm_{o,i}}$. В рамках данного выражения j представляет собой элементы, которые используются внутри веб-страницы. Значение $T_{elm_{o,i}}$ является весом сложности, который присвоен элементу $elm_{o,i}$. В свою очередь он представляет собой дополнительный элемент (*CSS, JavaScript*), влияющий на время отклика. Таким образом, C_2 может быть определен как функция от elm_o, т. е. $C_2 = f(elm_o)$. На основе этого можно вывести следующее выражение для сложности контента веб-страниц $C_2 = f(n_o, z_{sp}, elm_o)$, которое может быть выражено в виде кортежа (a,b,c), где a, b и c – квантифицированные значения для $g_{obj}(n_o), g_{size}(z_{sp}), g_{assets}(elm_o)$

соответственно и, где: g_{obj} – вклад числа дополнительных объектов; g_{size} – вклад размера страницы; g_{assets} – вклад элементов (веб-ассеты).

При построении моделей учитывались фундаментальные результаты исследований в области анализа производительности клиент-серверных систем, где обоснована значимость метрик количества строк кода, сложности программных конструкций, обращений к базе данных, а также объема и структуры ресурсов. В разработанных моделях данные метрики формализованы в виде функций, отражающих их вклад во время отклика.

Разработанная методика анализа производительности клиент-серверных веб-приложений представляет собой интеграцию пяти взаимосвязанных компонентов: метода сбора данных о производительности клиент-серверных приложений на основе программного комплекса, двух моделей, механизма мониторинга и алгоритма.

Для ее реализации был создан алгоритм, который итерационно выполняет сбор и анализ данных, сравнивает текущую и предыдущую версии страниц с эталонными профилями и фиксирует превышение порогов, после чего выполняется разложение вкладов отдельных метрик в общие изменения (рисунок 4). Использование алгоритма позволяет выявлять причины роста времени отклика, формировать рекомендации на что разработчику стоит обратить внимание и обеспечивать снижение задержек при работе приложения. Для алгоритма, который лежит в основе методики, m представляет собой метрики, которые являются частью составной метрики: $M = \{m_1, m_2, ..., m_n\}$. Для алгоритма выполняется инициализация порогов τ для всех метрик модели в каждой соответствующей подвыборке условий, а также фиксируются параметры сравнения: агрегирующие функции $agg_m(x)$, выполняется сбор данных производительности и формируется текущий профиль страницы P_t, содержащий данные по метрикам модели. Для каждой метрики m вычисляется сводное текущее значение $x_t(m) = agg_m(P_t)$ и проверяется превышение заданного порога. Если его нет, то профиль сохраняется, и после этого алгоритм завершает работу. При наличии превышения происходит запуск проверки для данной подвыборки. Если ранее эталонный профиль не существовал, то он используется как эталон P_0 для последующих проверок. Однако, если это не первый запуск, то выполняется сбор данных, при котором повторно фиксируется P_t, анализируется веб-страница по модели ее генерации и по модели ее контента. После этого происходит анализ: $m = x_t(m) - Q_m$, где Q_m – эталонное значение метрики, с которым сравнивается текущее значение при анализе. Далее выполняется проверка порогов срабатывания в соответствии с порогами и правилами модели и регистрируются результаты проверки вместе с контекстом подвыборки. На основании полученных дельт и значений метрик сложности формируются рекомендации по улучшению. Далее алгоритм завершает цикл и готов к следующему запуску при неизменности определений. Методика и алгоритм были реализованы в программе (Свидетельство о регистрации программы для ЭВМ №2025665430 от 17.06.2025).

Метод сбора данных обеспечивает фиксацию ключевых метрик производительности и затем их последующую агрегацию в процессе работы веб-приложений. В рамках работы особое внимание уделяется метрике времени отклика. В свою очередь математическая модель генерации и контента позволяют веб-странице формализовать метрики веб-страницы, которые влияют на время отклика, тем самым позволяя количественно оценивать их воздействие на производительность. Мониторинг обеспечивает накопление и систематизацию данных о метриках производительности в реальных условиях.

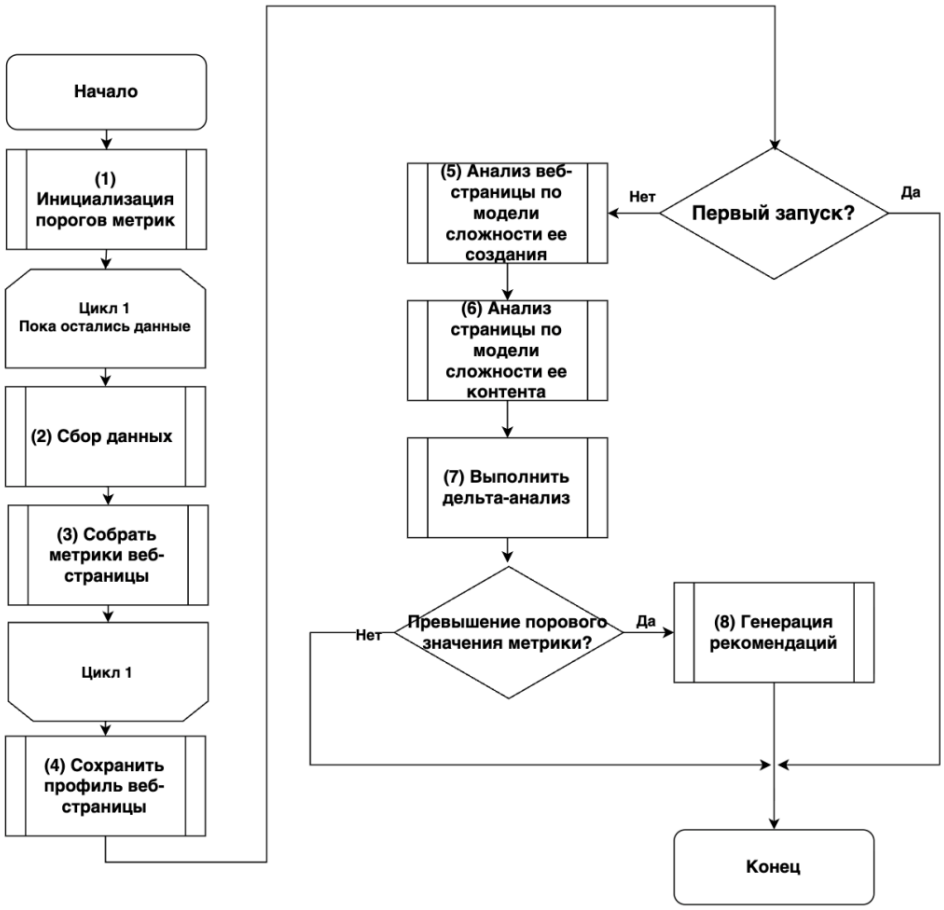

Рисунок 4 – Алгоритм выявления отклонений производительности веб-страниц

В рамках **третьей главы** был разработан программный комплекс, основанный на микросервисной архитектуре, и, который используется для сбора данных о производительности веб-приложений.

Диаграмма контекста C_4 показывает систему как ключевой элемент, который связывает таких пользователей как менеджер, разработчик и тестировщик с процессом сбора данных производительности сторонних приложений и их анализа. Сторонними веб-приложениями включают клиент-серверные веб-приложения и отдельно взятые клиентские, и серверные веб-приложения. Следует подчеркнуть, что они являются источниками данных, которые будут потребляться разработанным программным комплексом. Диаграмма контекста демонстрирует в каком окружении происходит работа с системой и какие основные взаимодействия выполняются над ней (рисунок 5).

Диаграмма контейнеров C_4 показывает архитектуру разворачиваемых единиц в рамках разработанной системы сбора данных о производительности клиент-серверных веб-приложений (рисунок 6). Взаимодействие пользователей с системой реализуется через веб-клиент, который представляет собой пользовательский интерфейс, позволяющий отображать и выгружать полученные данные. Клиентская часть системы отправляет запросы на сервер, который отвечает за обработку поступающих данных и дальнейшую их отправку к другим сервисам.

Для обеспечения сбора метрик непосредственно в сторонних веб-приложениях в систему интегрирована специализированная программная библиотека-агент. Так для клиентских веб-приложений это язык *JavaScript*. Данный компонент встраивается в клиентскую часть стороннего веб-приложения, осуществляет мониторинг ключевых показателей производительности и передает данные в серверную часть системы для последующего анализа. Функциональное взаимодействие между компонентами реализуется посредством *HTTP(S)*-запросов в формате *JavaScript Object Notation* или *JSON*.

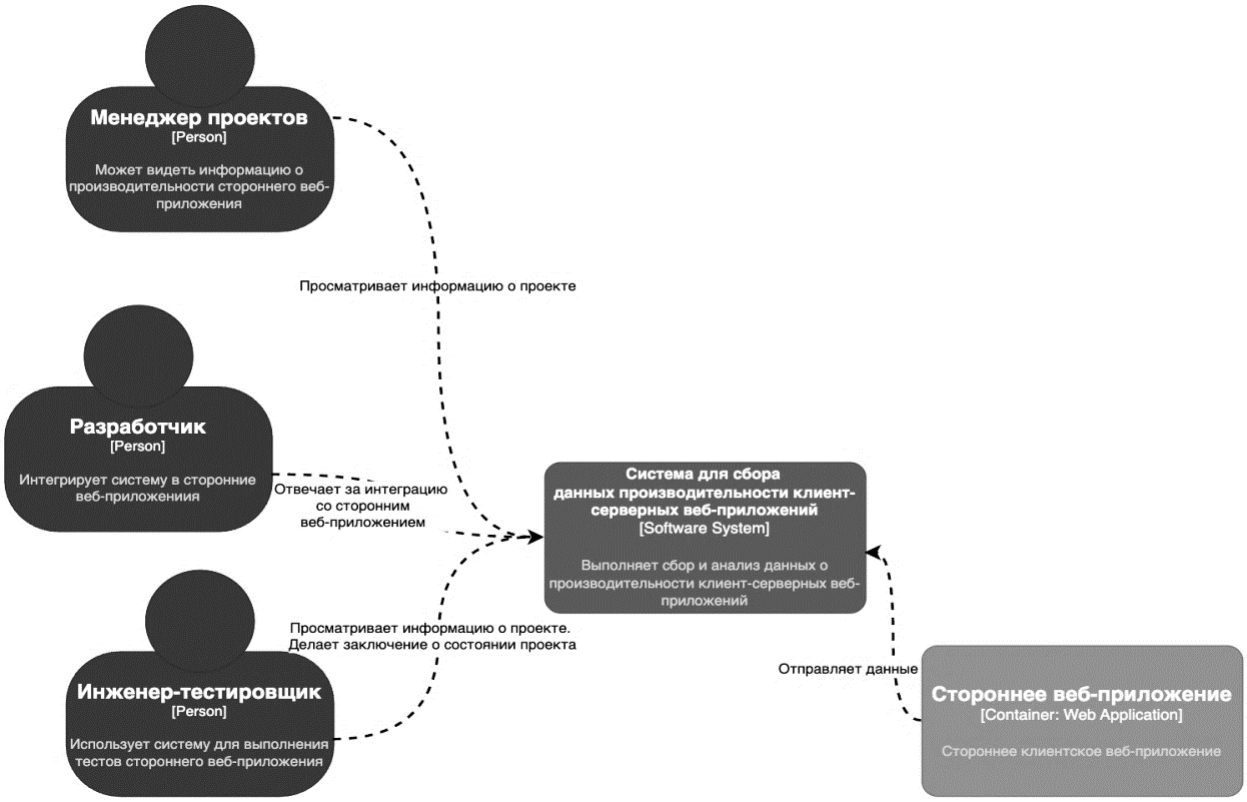

Рисунок 5 – Диаграмма контекста

Интегрированная библиотека-агент собирает данные и отправляет на сервер, обеспечивая централизованный анализ метрик. Разработчики несут ответственность за интеграцию библиотеки в сторонние веб-приложения, обеспечивая корректную передачу данных. Инженеры по тестированию используют систему для анализа состояния проекта и выявление отклонений в рамках времени отклика на каждой итерации разработки программного обеспечения, а менеджеры проектов получают доступ к результатам мониторинга для принятия управленческих решений.

Рисунок 6 – Диаграмма контейнеров

Диаграмма компонентов представлена на рисунке 7. Данная диаграмма детально описывает компоненты системы, без которых ее работа будет невозможна.

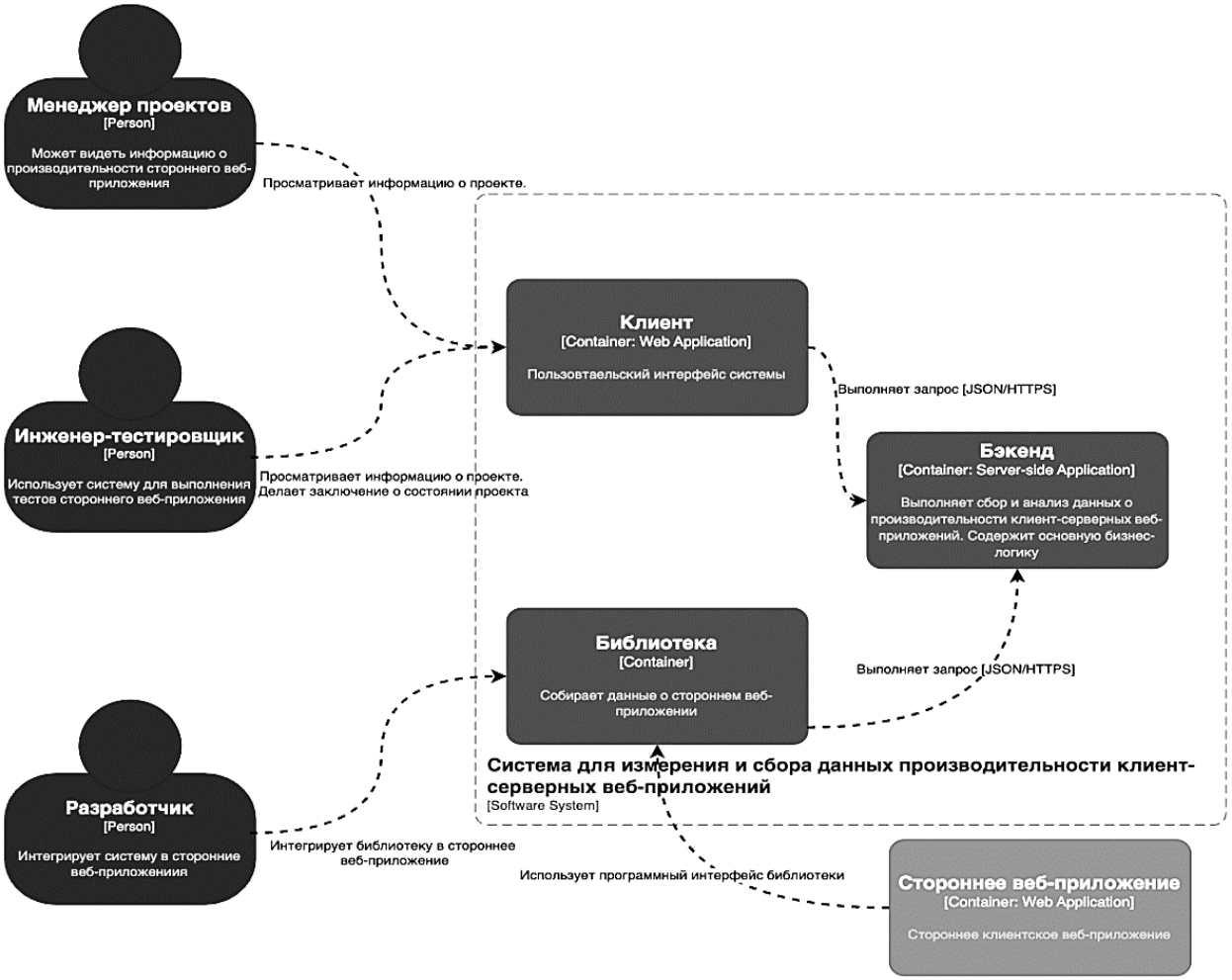

Рисунок 7 – Диаграмма компонентов

Веб-система для мониторинга веб-приложений является комплексной системой, состоящей из множества компонентов – микросервисов. Несмотря на то, что клиент не является отдельным сервисом, который отвечает за обработку данных, но он представляет собой основной пользовательский интерфейс приложения, через который происходит взаимодействие пользователя с остальными компонентами разработанной системы. Он дает пользователям возможность взаимодействия с ней, выполнять мониторинг метрик текущих приложений, которые отправляют данные.

Отдельная библиотека-агент позволяет собирать данные из сторонних приложений и пересылать их на сервер для дальнейшего хранения и анализа. Балансировщик отвечает за обработку входящих запросов и распределение нагрузки между несколькими экземплярами сервиса *REST API.* Причина использования балансировщика заключается в том, что на него поступает большое количество запросов и равномерное распределение нагрузки для нескольких экземпляров сервиса позволяет выдержать большой приток запросов. Такой подход к проектированию веб-систем позволяет обеспечить высокую доступность системы. *REST API* сервис содержит конечные точки, принимающие и отдающие данные конечному авторизованному пользователю. Также данный сервис позволяет управлять текущими приложениями и пользователями.

Очередь *Apache Kafka* содержит несколько топиков для общения между микросервисами. Это позволяет асинхронно передавать данные между различными частями системы, при этом делая их независимыми друг от друга. База данных *PostgreSQL* хранит информацию о пользователях, проектах и клиентах, в то время как *MongoDB* хранит информацию о метриках в виде временных рядов. Сервис уведомлений уведомляет пользователей, если их приложения превышают определенные пороговые значения в

рамках заданных метрик. Сервис генератор отчетов позволяет создавать отчеты на основе полученных данных. *Measurements bundler* обрабатывает и преобразует данные о метриках в формат необходимый для дальнейшего хранения в системе.

В системе мониторинга производительности веб-приложений компоненты взаимодействуют посредством строго определенных интерфейсов, обеспечивая непрерывный поток данных от этапа сбора до представления конечному пользователю. Клиентская часть взаимодействует с балансировщиком нагрузки *Nginx*. Его задачей является распределение входящих запросов между несколькими экземплярами *REST API*. Такой подход способствует равномерному распределению нагрузки, предотвращает перегрузку отдельных экземпляров сервиса и обеспечивает высокую отказоустойчивость системы. *REST API* осуществляет взаимодействие с двумя системами управления базами данных: реляционной базой данных *PostgreSQL*, предназначенной для хранения пользовательских учетных записей, проектов и настроек и нереляционной базой *MongoDB,* в которой сохраняются непосредственно данные о метриках производительности. Доступ к данным осуществляется через стандартные драйверы и протоколы, обеспечивая надежность и гибкость обработки информации. В свою очередь *REST API*, осуществляющий обработку входящих запросов, формирует сообщения, которые публикуются в брокер сообщений *Apache Kafka,* являющийся распределенной системой для асинхронной работы с ними. Его использование позволяет осуществлять асинхронную передачу данных и уменьшить связанность между сервисами. Полученные с помощью представленной системы данные могут быть в дальнейшем использованы специалистами в области анализа данных. Представленные диаграммы позволяют более наглядно описать те единицы, из которых состоит программный комплекс, который в свою очередь является неотъемлемой частью метода для сбора данных о производительности.

В четвертой главе представлены результаты работы программного комплекса, которые связаны с метриками тестовых веб-страниц, созданных для установки связи между разработанными теоретическими метриками, моделями и практическими данными, полученными в результате экспериментов. Каждая из формализованных характеристик веб-страницы (строки кода, сложность программных конструкций, количество дополнительных объектов, размер ресурсов, число соединений с базой данных и переменных *cookie*) была соотнесена с измеряемыми показателями производительности, фиксируемыми программным комплексом (время отклика, время рендеринга, время передачи данных). Проведенные эксперименты показали, что изменение теоретических метрик закономерно отражается на практических показателях, что подтверждает корректность моделей.

В качестве веб-браузера использовался *Google Chrome Browser* версии 133 с использованием *Next.js* фреймворка. Данная технология является наиболее популярным инструментом при разработке клиент-серверных веб-приложений и требует нетривиального подхода при анализе метрик генерации веб-страниц по причине использования *jsx*-кода. В данной работе они рассматривается, как и обычные строки кода.

В ходе исследования были созданы экспериментальные условия, чтобы минимизировать влияние внешних факторов на время отклика: веб-страницы обслуживались одним и тем же веб-сервером, доступ к страницам осуществлялся в рамках локальной сети.

Время отклика страницы R представляет собой функцию от сложности генерации веб-страницы C_1 и сложности контента C_2 и имеет следующий вид: $R = f(C_1, C_2)$, где значение времени отклика стремится к минимуму.

Для проведения экспериментов для изучения влияния метрики «Строки кода» был создан первый набор из четырех статических веб-страниц. Их отличие заключается в количестве строк кода.

Таблица 1 – Детали страниц

Параметр / Название страницы	Страница 1а	Страница 1б	Страница 1в	Страница 1г
Строки кода	104	14	104	1000
Размер (КБ)	2	2	5	12

Поскольку четыре страницы статичны, их C_1 является функцией от количества строк кода в форме $linescode \cdot w_1$, где $linescode$ обозначает количество строк кода, а w_1 – вес выполнения, который назначен этой метрике. С другой стороны, количество строк кода для страниц 1а, 1б, 1в и 1г составляет $104w_1$, $14w_1$, $104w_1$ и $1000w_1$ соответственно.

Для каждой страницы было зафиксировано более 100 значений времени рендеринга, времени отклика и время первого отображения. Таблица 2 показывает средние значения (μ) и стандартные отклонения (σ) для полученных значений, а рисунок 8 показывает распределение полученных значений времени.

Таблица 2 – Полученные временные показатели веб-страниц для сравнения влияния строк кода

Метрика / № страницы	Страница 1а (μ)	Страница 1а (σ)	Страница 1б (μ)	Страница 1б (σ)	Страница 1в (μ)	Страница 1в (σ)	Страница 1г (μ)	Страница 1г (σ)
Время рендеринга (мс)	15,8	0,58	15,2	0,48	16,1	0,57	31,3	0,48
Время отклика (мс)	31,9	0,32	31,2	0,42	47,2	0,42	62,4	0,52
Время до первого отображения (мс)	15,5	0,42	15,9	0,32	31,1	0,32	31,1	0,29

В таблице 3 имеются средние значения (μ) и стандартные отклонения (σ), полученные на основе результатов десяти значений.

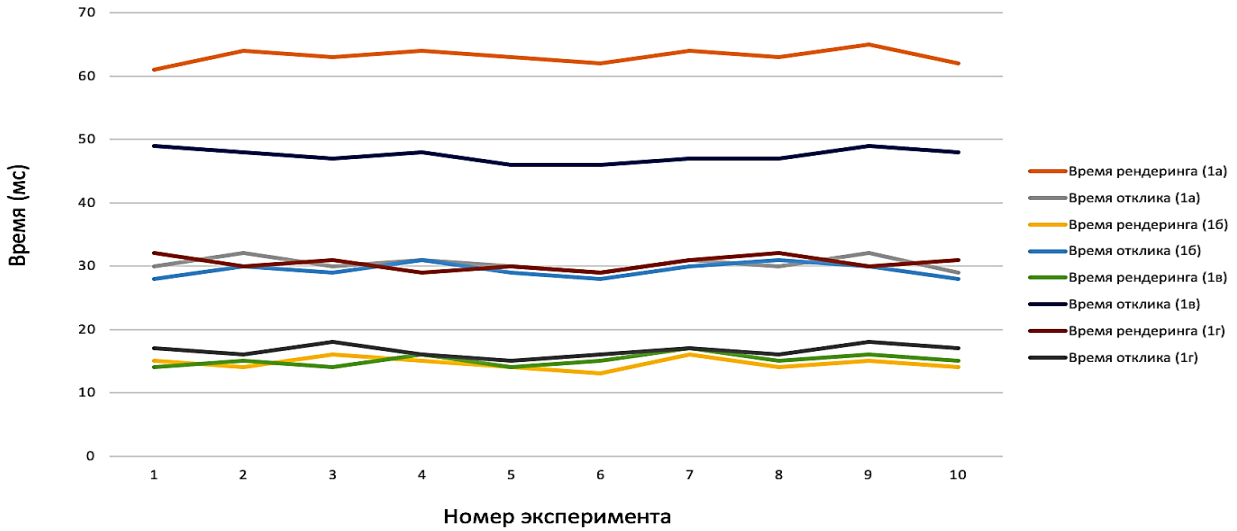

Рисунок 8 – Распределение собранных временных показателей веб-страниц для сравнения влияния количества строк кода

В процессе сбора данных условия на стороне сервера и сети оставались неизменными для обработки запроса и передачи ответа. В результате было обнаружено, что страница 1в имеет большее время отклика по отношению к другим страницам. В результате страница с большим количеством строк кода обладает увеличенным временем

отклика. Полученные данные подтверждают влияние этой метрики на общее время отклика R веб-страницы, соответствуя выражению $R = f(linecode)$.

Второй набор содержит три статических веб-страницы, отличие которых заключается в разном количестве дополнительных объектов. В данном случае это *JPEG*-изображения. Первая страница 2а содержит 4 дополнительных объекта, страница 2б содержит 10 дополнительных объектов, а страница 2в содержит 13 дополнительных объектов. Все изображения обладают одинаковым разрешением (1024 × 1024 пикселей), а их общий размер на каждой странице идентичен и составляет 2500 КБ. После получения данных в виде 100 значений было выявлено, что время отклика веб-страницы увеличивается пропорционально количеству дополнительных объектов (таблица 3).

Таблица 3 – Полученные данные веб-страниц для сравнения влияния количества дополнительных объектов

Метрика / № страницы	Страница 2а (μ)	Страница 2а (σ)	Страница 2б (μ)	Страница 2б (σ)	Страница 2в (μ)	Страница 2в (σ)
Время рендеринга (мс)	920	24	1474	33	1992	37
Время отклика (мс)	953	25	1525	25	2058	35
Время до первого отображения (мс)	32	6	49	16	63	19

Данное поведение объясняется тем, что увеличение количества дополнительных объектов влияет на браузер. По этой причине ему требуется больше времени на их отображение. Более того, возрастает количество запросов между клиентом и сервером.

В следующем наборе экспериментов для набора страниц 2а, 2б, 2в использовались дополнительные объекты с разным размером 1511, 1998 и 2509 килобайт соответственно. После анализа полученных значений было выявлено, что с увеличением общего размера страницы sz_p, возрастает и ее время отклика R. Описанное поведение обусловлено увеличением объема данных, передаваемых по сети от сервера к клиенту.

Интеграция файлов *JavaScript* и *CSS* может значительно увеличить время отклика веб-страницы. Для оценки влияния данного аспекта в рамках эксперимента была модифицирована страница 1а из первого набора. На ее основе были созданы две версии: 1д и 1е. Вариант 1д включает встроенный *JavaScript*-код, который выводит текущие дату и время в заголовке страницы. Для версии страницы 1е применены каскадные таблицы стилей, задающие цвет фона и шрифта. В ходе эксперимента было зафиксировано увеличение времени отклика веб-страниц, содержащих код и таблицы стилей, что связано с добавлением дополнительных elm_o элементов, которые в свою очередь на процесс рендеринга и загрузки ресурсов. Следовательно, данные элементы влияют на время отклика, что можно выразить как $R = f(elm_o)$. Полученные средние значения и стандартные отклонения представлены в таблице 4.

Таблица 4 – Полученные временные показатели веб-страниц для изучения влияния *JavaScript* и *CSS*

Метрика / № страницы	Страница 1а (μ)	Страница 1а (σ)	Страница 1д (μ)	Страница 1д (σ)	Страница 1е (μ)	Страница 1е (σ)
Время рендеринга (мс)	15,9	0,57	16,1	0,47	16,2	0,62
Время отклика (мс)	31,9	0,30	46,9	0,30	47,0	0,47
Время до первого отображения (мс)	15,5	0,42	30,9	0,31	30,8	0,41

Чтобы определить влияние сложности программных конструкций на время отклика были разработаны три веб-страницы: 4а, 4б, 4в. Каждая из веб-страниц имеет 3, 4 и 3

программных конструкции соответственно. Каждая из страниц имеет три компонента пользовательских интерфейса – раскрывающихся списка. Они имеют три поля: день, месяц и год. Отображение содержимого каждого из данных полей происходит с использованием трех циклов *for*. Два из них используются для создания списков дня и месяца. Они остаются идентичными на всех трех страницах. Отличие присутствует только в третьем цикле, который отвечает за значение года. Для страницы 4а, имеющей 100 элементов в выпадающем списке, используется один цикл *for*. Сложность данного цикла в рамках *Big-O* нотации оценивается как *O(n)*, где n = 100. Страница 4б содержит 100 элементов в выпадающем списке, где сложность цикла оценивается как n = 100. Однако в данном случае присутствует дополнительный вложенный цикл *while*, который выполняет 450 повторений, выполняя операцию сложения на каждой итерации. Страница 4в содержит в выпадающем списке 1000 элементов. Используется один цикл *for*. Сложность является *O(n)*, где n = 1000. Более детальные описания программных конструкций представлены в таблице 5.

Таблица 5 – Описание программных конструкций

Страница	Цикл	$f(n)$	n	*linecode*	$f(n)\cdot linecode$
Страница 4а	Цикл 1	n	31	3	92
	Цикл 2	n	12	3	35
	Цикл 3	n	100	4	400
Страница 4б	Цикл 1	n	31	3	93
	Цикл 2	n	12	3	36
	Цикл 3	n	100	5	500
	Цикл 4	n·m	100, 450	2	100000
Страница 4в	Цикл 1	n	31	3	93
	Цикл 2	n	12	3	36
	Цикл 3	n	450	4	2000

Полученные в рамках сбора данных результаты показывают, что страница 4в, в отличие от страниц 4а и 4б, имеет более сложные синтаксические конструкции, что приводит к увеличению времени рендеринга (таблица 6).

Таблица 6 – Полученные временные показатели страниц с различной сложностью программных конструкций

Метрика / № страницы	Страница 4а (μ)	Страница 4а (σ)	Страница 4б (μ)	Страница 4б (σ)	Страница 4в (μ)	Страница 4в (σ)
Время рендеринга (мс)	71,1	7	71,1	7	85	8
Время отклика (мс)	374	36	473	30	596	21
Время до первого отображения (мс)	302	41	401	30	509	16

Кроме того, количество итераций для цикла *for*, используемого на странице 4в, было больше, что существенно увеличило время обработки на сервере по сравнению с двумя другими страницами. В результате чего наблюдается более длительная задержка перед началом отображения страницы в браузере.

Метрика «состояние приложения» является количественным показателем, который отражает совокупное влияние таких внутренних факторов веб-приложения как обращения к базе данных, работа с сессионными переменными на время отклика. Для анализа метрики были созданы три веб страницы: 5а, 5б, 5в, которые генерируются динамически. Для минимизации влияния передачи по сети на время отклика контент страниц использовалось только текстовое

содержимое. Для создания страницы 5а устанавливается соединения с базой данных и выполняется запрос к таблице. Полученная страница имеет размер в 5 КБ и имеет список из 45 элементов. Для страницы 5б устанавливается соединение с базой данных и выполняются два запроса к двум таблицам. Полученная страница имеет общий размер 6 КБ. Она содержит два раскрывающихся списка, содержащих по 45 и 25 элементов соответственно. Отличие между страницами 5б и 5а заключается в дополнительных запросах к базе данных, увеличенным количеством строк кода и дополнительными программными конструкциями, которые необходимы для обработки возвращаемых данных, а также увеличенным объемом конечной страницы, которая отображается в браузере. Для случая создания страницы 5в устанавливаются 2 подключения к базе данных и выполняются два запроса к двум таблицам. Полученная страница имеет размер 8 КБ, а также список, содержащий 45 элементов и список, содержащий 25 элементов. Запросы и таблицы взяты из примера страницы 5б для применения в рамках полученных данных для страницы 5в. Данное условие позволяет обеспечить одинаковый контент для страницы 5в. В таблице 7 приведены средние значения и стандартные отклонения, полученные на основе результатов полученных данных.

Таблица 7 – Сравнение состояний доступа к базе данных

Метрика / № страницы	Страница 5а (μ)	Страница 5а (σ)	Страница 5б (μ)	Страница 5б (σ)	Страница 5в (μ)	Страница 5в (σ)
Время рендеринга (мс)	43	6	69	8	69	7
Время отклика (мс)	2730	40	2872	61	3091	59
Время до первого отображения (мс)	2685	41	2801	62	3021	54

Время отклика для страниц 5б и 5в занимает больше времени, если сравнивать со страницей 5а, что можно обосновать размерами 5б и 5в. Чем больше размер, тем больше время отклика. Однако стоит отметить, что 5б и 5в имеют схожее время рендеринга, что подтверждает их идентичность в контексте исследуемой метрики. Дельта среднего времени отклика для страниц 5а и 5б составляет 142 мс, а для страницы 5б и 5в – 219 мс. Полученные данные дают понимание того, что установление соединений с базой данных занимает больше времени, чем выполнение запроса.

Для изучения влияния данных *cookie* на время отклика были созданы четыре веб-страницы. Страница 6а не имеет *cookie*, что значит хранение данных не выполняется. Другая страница 6б имеет содержит 17 переменных сессии, которые создаются сервером при выполнении запроса на получение страницы и передаются на браузер клиента, а страница 6в выполняет операцию по извлечению 20 переменных сессии *cookie*, которые связаны со страницей 6б. Страница имеет логику, которая позволяет модифицировать и затем отправлять обновленные данные. Страница 6г является копией 6б, однако данные *cookie* в данном случае содержат 45 переменных сессии вместо 17. В результате страницы 6б и 6г обеспечивают условия, при которых создаются дополнительные переменные в *cookie.*

Таким образом, страницы 6б и 6г моделируют сценарий, когда *cookie* создаются на сервере и передаются клиенту, тогда как страница 6в обеспечивает процесс двустороннего обмена между клиентом и сервером данными *cookie*, включая манипуляции над этими данными. Информация о страницах представлена в таблице 8.

Таблица 8 – Информация о страницах с разными состояниями *cookies*

Параметр / Название страницы	Страница 6а	Страница 6б	Страница 6в	Страница 6г
Строки кода	23	30	35	35
Программные конструкции	0	1	1	1
Состояние *cookies* (переменные)	0	17	20	45
Количество дополнительных объектов	0	0	0	0
Размер (КБ)	1	1	1	1
Количество элементов (JS/CSS)	0	0	0	0

Для каждой из разработанных страниц было зафиксировано 100 значений времени отклика. Таблица 9 содержит средние значения и стандартные отклонения для полученных данных.

Таблица 9 – Влияние *cookies*

Метрика / № страницы	Страница 6а (μ)	Страница 6а (σ)	Страница 6б (μ)	Страница 6б (σ)	Страница 6в (μ)	Страница 6в (σ)	Страница 6г (μ)	Страница 6г (σ)
Время рендеринга (мс)	17	0,54	17	0,42	17	0,54	17	0,53
Время отклика(мс)	32	0,33	32	0,53	47	0,43	32	0,52
Время до первого отображения (мс)	17	0,51	17	0,48	34	0,47	17	0,33

Веб-страницы, обрабатывающие данные *cookie*, имеют увеличенное время отклика. Для оценки влияния данного фактора были созданы два тестовых сценария: передача и их модификации. Результаты показали, что данные сессии *cookie* приводят к росту времени отклика за счет добавления дополнительной информации в обмен между клиентом и сервером.

Таким образом, общее время отклика веб-страницы может быть представлено в виде функции, зависящей от характеристик *cookie*: количество, размер, а также тип выполняемых над ними операций (чтение, передача, модификация).

В заключении работы сформулированы основные выводы и перечислены результаты, полученные в ходе исследования по разработке технических решений для сбора и анализа данных о производительности клиент-серверных веб-приложений. Выполненные эксперименты показали, что варьирование каждой из метрик приводит к закономерным изменениям показателей, фиксируемых программным комплексом. Так рост числа строк кода и усложнение синтаксических конструкций приводят к увеличению времени генерации страницы на сервере. Увеличение числа дополнительных объектов и их размеров прямо коррелировало с ростом времени рендеринга и времени отклика на клиентской стороне, увеличение количества подключений к базе данных и операций с *cookie* сопровождалось ростом совокупного времени отклика.

Проведен обзор существующих систем, методов анализа и оценки производительности клиент-серверных веб-приложений

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

В ходе диссертационного исследования получены следующие результаты, имеющие существенное значение для развития области информационных технологий.

1. Проведен анализ систем и методов анализа производительности клиент-серверных веб-приложений. Выявлены ограничения существующих методов и недочеты схожих технических решений. Полученная в результате обзора и анализа информация послужила созданию требований для разработанного программного комплекса.
2. Разработана математическая модель генерации веб-страницы, которая учитывает влияние таких метрик как количество строк кода, сложность программных конструкций и состояние приложения (количество подключений к базе данных, количество переменных *cookie*) на время отклика.
3. Разработана математическая модель контента веб-страницы, описывающая влияние таких метрик контента как общий размер *HTML*-документа, количество дополнительных объектов, размер и количество *JavaScript* и *CSS*-ресурсов на время отклика. Модель позволяет определять влияние факторов, связанных с контентом веб-страницы, на время отклика.
4. Разработан метод сбора данных о производительности клиент-серверных веб-приложений, основанный на программном комплексе и использующий микросервисную архитектуру для автоматизированного сбора, анализа и визуализации данных, результатом внедрения которого стало улучшение качества оценки производительности. Использование

программного комплекса обеспечивает снижение времени отклика веб‑страниц в среднем на 30% по сравнению с традиционным методом, основанным на прокси‑сервере.

5. Предложена методика анализа производительности веб-страниц клиент-серверных приложений, основанная на интеграции метода сбора данных производительности на основе программного комплекса, математических моделей генерации и контента веб-страниц с использованием алгоритма классификации и выявления закономерностей, что позволяет установить взаимосвязь между характеристиками веб-страниц и их временем отклика, а также обеспечить верифицируемую оценку производительности.

6. Проведено экспериментальное исследование, подтвердившее применимость предложенных методов, моделей и методики для анализа факторов, влияющих на производительность клиент-серверных приложений, и обеспечивающих поддержку на этапе их разработки и в процессе эксплуатации.

7. Разработан программный комплекс для сбора и анализа данных о производительности клиент-серверных приложений, применение которого в АО Радиозавод позволило снизить время на диагностику проблем на 13,7%.

СПИСОК РАБОТ, ОПУБЛИКОВАННЫХ ПО ТЕМЕ ДИССЕРТАЦИИ

Публикации в изданиях из перечня ВАК Минобрнауки России

1. Максимов, Я.А. Обзор современных программных решений в области измерения производительности клиентской части веб-приложений / Я.А. Максимов, А.И. Мартышкин // Современные наукоемкие технологии. – 2021. – № 12-2. – С. 348-354.

2. Максимов, Я.А. Влияние интернет-рекламы на производительность клиентской части веб-приложений / Я.А. Максимов, А.И. Мартышкин // Современные наукоемкие технологии. – 2022. – № 11. – С. 52-56.

3. Максимов, Я.А. Как работает выделение памяти на примере браузерного движка V8 / Я.А. Максимов, А.И. Мартышкин // Современные наукоемкие технологии. – 2022. – № 6. – С. 40-46.

4. Максимов, Я.А. Метод расчета производительности веб-приложения / И.С. Пышкина, Я.А. Максимов, А.И. Мартышкин // XXI век: итоги прошлого и проблемы настоящего плюс. – 2024. – Т. 13. – № 2(66). – С. 43-48.

5. Максимов, Я.А. Подход для создания мобильных приложений, получающих обновления независимо от магазинов приложений / Я.А. Максимов, А.И. Мартышкин, Е.Д. Касаткина, М.И. Панфилова // XXI век: итоги прошлого и проблемы настоящего плюс. – 2025 – Т. 14 – № 2(70). – С. 144-148. – EDN: YJPBUN.

Публикации в изданиях, индексируемых базой Scopus

6. Maximov, Y.A. Fault Prediction in Server-Side of Web-Applications by Using a Software System and Long Short-Term Memory Neural Networks / Y.A. Maximov, A.I. Martyshkin // 2025 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), Sochi, Russian Federation, 2025, pp. 1023-1027, doi: 10.1109/ICIEAM65163.2025.11028507.

Публикации в прочих изданиях

7. Максимов, Я.А. Машины состояний в современном JavaScript / Я.А. Максимов, Е.А. Гудкова // Современные информационные технологии. – 2020. – № 31 (31). – С. 42-45.

8. Максимов, Я.А. Развертывание современных веб-приложений / Я.А. Максимов, Е.А. Гудкова // Современные информационные технологии. – 2020. – № 32 (32). – С. 69-71.

9. Максимов, Я.А. Анализ и выбор метрик оценки производительности клиентской части веб-приложений / Я.А. Максимов, А.И. Мартышкин // Актуальные вопросы современной науки: теория и практика научных исследований: сборник научных трудов всероссийской научно-практической конференции. Пенза, 2021. – С. 32-35.

10. Максимов, Я.А. Проблемы измерения производительности в современных клиент-серверных веб-приложениях / А.И. Мартышкин, Я.А. Максимов // Современные информационные технологии. – 2023. – № 37 (37). – С. 105-109.

11. Максимов, Я.А. Кибербезопасность технологии интернет вещей / Я.А. Максимов, А.И. Мартышкин // Современные информационные технологии. – 2024. – № 39 (39). – С. 185-189.

12. Максимов, Я.А. Возможности оптимизации производительности веб-приложений / Я.А. Максимов, А.И. Мартышкин // Современные информационные технологии. – 2024. – № 39 (39). – С. 189-192.

13. Максимов, Я.А. Подход к сбору данных о производительности на серверной части веб-приложений / Я.А. Максимов // Новые научные исследования и разработки 2025: сборник статей Международной научно-практической конференции. Пенза, 2025. – С. 21-23.

14. Максимов, Я.А. Использование метрик DevOps для повышения качества программного обеспечения / Я.А. Максимов // Молодые учёные России: сборник статей XXIV Всероссийской научно-практической конференции. Пенза, 2025. – С. 56-58.

15. Максимов, Я.А. Подход к прогнозированию сбоев на серверах на основе нейронной сети долгой краткосрочной памяти / Я.А. Максимов // Тенденции развития науки и образования. – 2025. – № 122. – Ч. 5. – С. 105-109.

Свидетельства о государственной регистрации программ для ЭВМ

16. Свидетельство о государственной регистрации программ для ЭВМ № 2025664004. Программа для сбора метрик в браузерных приложениях // Максимов Я.А., Мартышкин А.И. Пензенский государственный технологический университет. 02.06.2025.

17. Свидетельство о государственной регистрации программ для ЭВМ № 2025665430. Программа для измерения метрик производительности клиент-серверных систем // Максимов Я.А., Мартышкин А.И. Пензенский государственный технологический университет. 17.06.2025.

МАКСИМОВ ЯРОСЛАВ АЛЕКСАНДРОВИЧ

МЕТОДЫ, МОДЕЛИ И МЕТРИКИ СБОРА ДАННЫХ О ПРОИЗВОДИТЕЛЬНОСТИ КЛИЕНТ-СЕРВЕРНЫХ ВЕБ-ПРИЛОЖЕНИЙ

Специальность 2.3.8. – Информатика и информационные процессы
(технические науки)

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Подписано в печать __.10.2025. Формат 60x84 $^{1}/_{16}$
Бумага офсетная. Печать цифровая. Усл. печ. л. 1,2.
Тираж 100 экз.
Пензенский государственный технологический университет.
440039, Россия, г. Пенза, пр. Байдукова/ул. Гагарина, $1^{а}$/11