Бондаренко О.С., Смирнов А.А., Вдовин В.С. Сравнительный анализ современных методов рендеринга ВЕБ-приложений и их влияния на производительность //Международный журнал информационных технологий и энергоэффективности. – 2024. – Т. 9 № 6(44) с. 113–121

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.4'2

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СОВРЕМЕННЫХ МЕТОДОВ РЕНДЕРИНГА ВЕБ-ПРИЛОЖЕНИЙ И ИХ ВЛИЯНИЯ НА ПРОИЗВОДИТЕЛЬНОСТЬ

[1]Бондаренко О.С., [2]Смирнов А.А., [3]Вдовин В.С.
ФГАОУ ВО "НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ (ИТМО)», Санкт-Петербург, Россия (197101, город Санкт-Петербург, Кронверкский пр-кт, д. 49 литер а), e-mail: [1]rancerenly@gmail.com, [2]smirnov.andrew.1999@yandex.ru, [3]me@vladislavvdovin.ru

Постоянное развитие современных технологий сопровождается увеличением потребляемых ресурсов для их использования. Рост связан с непрерывным созданием нового контента и способов взаимодействия с ним, а также развитием используемых технологий и подходов к разработке продукта. Множество систем переходят на новый формат использования – в веб-среду, предпочитая повышать простоту взаимодействия с программой, так как сайт не требует установки дополнительного программного обеспечения и постоянных обновлений со стороны клиента, однако реализация подобной системы включает в себя большие наборы данных, их постоянную обработку, тем самым утяжеляя и усложняя веб-приложение.

Наращивание функционала в приложении приводит к необходимости оптимизировать внутренние процессы работы приложения путем анализа и адаптации техник рендеринга и отслеживания событий в веб-приложении.

С учетом развития современных технологий и способов размещения контента, стоимость ресурса, как и его вес, увеличиваются, в связи с чем появляется запрос на оптимизацию существующих подходов к разработке веб-приложений с целью получить тот же ресурс, но работающий быстрее и эффективнее в процессе взаимодействия с системой.

Актуальность работы обусловлена существованием проблемы оптимизации рендеринга веб-приложений является, так как эффективность работоспособности системы определяет уровень вовлеченности пользователей, что поднимает показатели конверсии сайтов.

Ключевые слова: рендеринг, веб-приложение, фронтенд, производительность, JavaScript.

COMPARATIVE ANALYSIS OF MODERN WEB APPLICATION RENDERING METHODS AND THEIR IMPACT ON PERFORMANCE

[1] Bondarenko O.S., [2] **Smirnov A.A.**, [3] **Vdovin V.S.**
NATIONAL RESEARCH UNIVERSITY OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS (ITMO), St. Petersburg, Russia (197101, St. Petersburg, Kronverkskiy pr-kt, 49), e-mail: [1]rancerenly@gmail.com, [2]smirnov.andrew.1999@yandex.ru, [3]me@vladislavvdovin.ru

The constant development of modern technologies is accompanied by an increase in the resources consumed for their use. Growth is associated with the continuous creation of new content and ways to interact with it, as well as the development of technologies used and approaches to product development. Many systems are switching to a new format of use – to the web environment, preferring to increase the ease of interaction with the program, since the site does not require the installation of additional software and constant updates from the client, however, the

implementation of such a system includes large datasets, their constant processing, thereby weighing down and complicating the web application.
Increasing the functionality in the application leads to the need to optimize the internal processes of the application by analyzing and adapting rendering techniques and tracking events in the web application.
Taking into account the development of modern technologies and methods of content placement, the cost of the resource, as well as its weight, are increasing, and therefore there is a request to optimize existing approaches to developing web applications in order to get the same resource, but working faster and more efficiently in the process of interacting with the system.
The relevance of the work is due to the existence of the problem of optimizing the rendering of web applications, since the efficiency of the system determines the level of user engagement, which raises the conversion rates of sites.

Keywords: rendering, web application, frontend, performance, JavaScript.

В работе «Сравнительный анализ популярных JavaScript фронтенд решений» Еремин М. В. анализирует современные фронтенд-фреймворки.[2] Начиная с Vue, автор определяет его как подходящий для одностраничных приложений и рендеринга на стороне сервера. Он отмечает высокую производительность и гибкость решения, однако риск чрезмерной гибкости относит к недостаткам, так как такое качество является негативным в рамках разработки большого и сложного проекта. Гибкость как негативное качество автор определяет в связи с большим количеством вариантов построения решения с использованием данного инструмента, это объясняется отсутствием строгой архитектуры проекта.

Касательно фреймворка Angular, стоит отметить, что есть две версии данного фреймворка: Angular и AngularJS. Первый представляет из себе переписанную версию старого AngularJS, включающую в себя множество улучшений и применение TypeScript как основного языка при разработке. В отличие от AngularJS, который полностью базируется на JavaScript.

К достоинствам фреймворка Angular Еремин М.В. относит производительность сервера и связывание данных. Механизм связывания данных в Angular позволяет отслеживать изменения в реальном времени, которые автоматически отображаются в модели, а производительность сервера улучшает производительность приложения в целом. К недостаткам использования этого фреймворка относит сложность архитектуры и большое количество файлов, связанных друг с другом, что объясняется множеством зависимостей в одном компоненте. Однако Толстых М.А. в статье отмечает, что строгая архитектура и высокая масштабируемость фреймворка Angular относится к достоинствам использования данного решения для реализации крупных и сложных систем в сегменте корпоративной разработки.

Толстых М.А. в своей работе определяет ReactJS как инструмент для быстрой разработки, в связи с низким порогом входа. Библиотека используется для создания динамических пользовательских интерфейсов, отличительным механизмом при изменении данных является виртуальный DOM, позволяющий эффективно управлять элементами на странице.

Последний, рассмотренный Ереминым М.В., популярный фреймворк – Svelte – обладает высокой производительностью у пользователя, так как работает на этапе сборки приложения, преобразуя код в стандартный JavaScript код, а также отличается отсутствием необходимости использовать виртуальный DOM.

Так, можно сравнить проанализированные решения по следующим критериям в Таблице 1.

Таблица 1 – Сравнение фронтенд-фреймворков по основным критериям

Критерий	VueJS	Angular	React	Svelte
Взаимодействие с DOM	Построение vDOM и сравнение с DOM	Incremental DOM	Построение vDOM и сравнение с DOM	Точечное обновление DOM
Архитектура	MVVM (officially)	MVVM-based	MVVM-based	MVVM-based
Масштаб проекта	Средний, небольшой	Крупный, средний	Средний, небольшой	Средний, небольшой

Авторы статьи «Рендеринг и его влияние на архитектуру веб-приложений» Заманов Е.А., Дутова Е.А. и Селецкая Н.Г., выделяют три основных подхода к рендерингу веб-приложений [3]:

1. Рендеринг на стороне клиента (Client-site rendering, CSR).
2. Рендеринг на стороне сервера (Server-side rendering, SSR).
3. Регидрация.

Суть первого подхода (CSR) заключается в отправлении сервером клиенту пустой HTML-страницы с некоторым набором скриптов, контент которой отображается в браузере при помощи JavaScript с использованием ресурсов сервера в дальнейшем за получением исходных данных.

К недостаткам можно отнести проблемы с SEO, высокая нагрузка на клиент, а также снижение производительности при масштабировании приложения.

Преимуществами такого подхода можно считать распределение нагрузки на веб-приложение между всеми клиентами и упрощение серверной части веб-приложение, как следствие, снижение затрат на поддержку ресурса.

Для решения проблемы с оптимальным отображением сайтов для поисковых систем является регидрация. Эта технология совмещает в себе подход к рендерингу как серверный, так и клиентский: с использованием веб-фреймворка на стороне клиента происходит преобразование статического DOM, полученного с сервера, в динамический, дополняя его функциональными элементами для корректной работы веб-приложения.

Фреймворками, которые относятся к подходу CSR, являются Vue, React, Angular и Svelte. Приложение в перечисленных фреймворках строится в виде дерева компонентов, каждый из компонентов описывает подмножество DOM, однако процесс взаимодействия с DOM отличается. В связи с ветвлением структуры компонентов, каждый компонент-родитель может разделять свое состояние с компонентами-потомками. Привязка данных реализована в одностороннем порядке из состояния приложения в пользовательский интерфейс, что гарантирует стабильность состояния приложения во время цикла рендеринга при манипуляции с DOM.

В статье «Современные фреймворки для разработки web-приложений» Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. отмечают различия в подходе к перерисовке DOM между React и Vue [4]. В ReactJS изменение состояния компонента приводит к изменению всего поддерева этого компонента, представление этого процесса отображено на Рисунке 1.

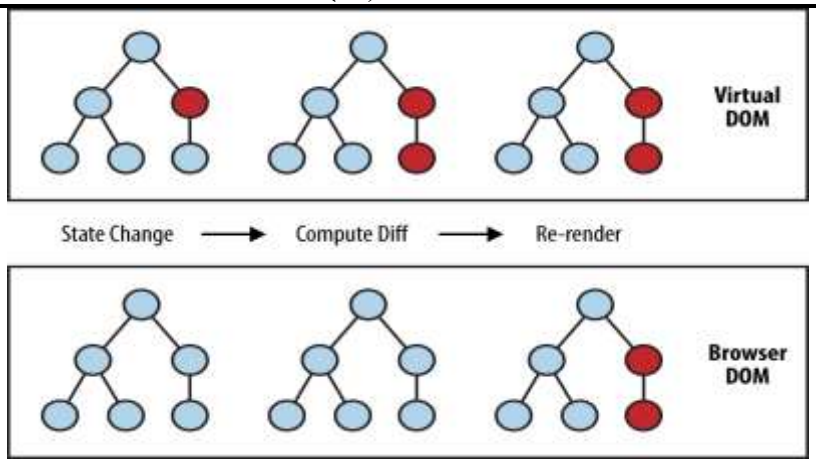

Рисунок 1 – Схема перерисовки DOM ReactJS

Vue реализует изменения точечно, автоматически отслеживая зависимости компонентов, что положительно сказывается на производительности приложения. Схема работы обновления DOM у VueJS представлена на Рисунке 2.

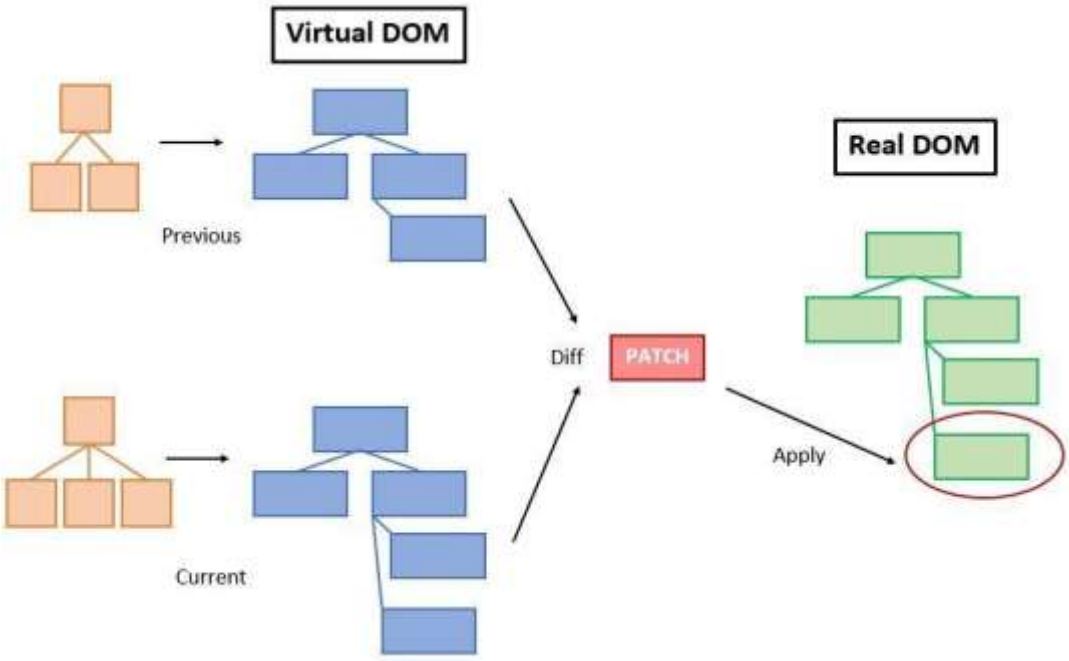

Рисунок 2 – Схема перерисовки DOM VueJS

К.Ю. Бетеев и Г.В. Муратова для определения механизмов эффективной перерисовки DOM ссылаются на книги Томаса Марка «React в действии» и Чиннатамби К. «Изучаем React», выделив следующие механизмы:

1. Использование паттерна проектирования «Наблюдатель».
2. Пакетное (batch) обновление DOM.
3. Алгоритм поиска различий с линейной сложностью O(n).

Паттерн Observer в React.js реализован при помощи распределения элементов в интерфейсе в виде компонентов, имеющих состояние (родительское или собственное). В

момент изменения состояния, запускается цепочка сравнения и перерисовки интерфейса в DOM.

Пакетное обновление представляет из себя сбор изменений, перерисовку которых можно отложить до завершения манипуляции с состоянием, позволяя снизив потерю эффективности в случае постоянных обновлений состояния компонента, таким образом, произведя процесс перерисовки единожды.

Поиск различий с линейной сложностью позволяет сравнивать деревья в vDOM перед отрисовкой в браузере в моменте согласования элементов.

Авторы статьи подмечают, что эффективная работа с DOM возможна с использованием различных практик, не требующих внедрения vDOM. Такие фреймворки как Angular и Svelte не используют данную концепцию, ориентируясь на точечных изменениях в DOM для эффективного обновления.

Как и в vDOM, каждый компонент определяет набор узлов DOM, которые должны быть отображены экземпляром компонента. В отличие от vDOM, здесь нет отдельного шага, на котором вычисляются все необходимые изменения пользовательского интерфейса.

Рендеринг заключается в прохождении по дереву компонентов, выполнении проверок привязок данных, чтобы увидеть, какие из них изменились и применении изменений к DOM для каждой найденной привязки.

Статья «Исследование методологии оптимизации рендеринга компонентов на примере Svelte», написанная Летоном Г., Глазько П.Е., описывает механизмы оптимизации рендеринга, выделяя одним из ключевых – уменьшение конечного размера веб-приложения.[5] Авторы статьи отмечают, что операции с DOM сильно влияют на производительность веб-приложений, соответственно, оптимальным решением для улучшения производительности является уменьшение взаимодействия с DOM.

Авторы статьи оценили производительность такой реализации решения и взаимодействия с DOM путем сравнения метрики FID с выполнением аналогичного функционала в React-приложении. Проанализировав и оценив работу двух приложений на основании выбранных метрик, Летон Г. и Глазько П.Е. пришли к следующему выводу: показатели приложения на Svelte выше примерно в полтора раза, компактность приложения. 24 Кб против 340 Кб в React отличается преимуществом использования Svelte вместо ReactJS, в случае отсутствия механизмов оптимизации в React-приложении разрыв в производительности будет только расти.

Данная статья позволяет сделать вывод, что использование механизмов компиляции кода в JavaScript при помощи компилятора-Svelte относит этот фреймворк к разряду более производительных на основании оценки метрик исследования.

Помимо рендеринга на клиентской стороне необходимо исследовать процесс серверного рендеринга веб-приложений, что позволит сравнить эффективность и производительность двух противоположных инструментов к реализации веб-сайтов.

SSR представляет возможность рендеринга двумя способами:

1. Статический – возвращение подготовленной HTML-страницы, которая была сгенерирована на этапе сборки сайта.

2. Контент по запросу – после запроса происходит обращение к базе данных, вследствие чего с помощью шаблонизатора происходит генерация страницы и отправка разметки в браузер.

Стоит отметить, что такой подход ведет к удорожанию стоимости разработки в связи с усложнением архитектуры приложения и высокой нагрузки на сервер, однако улучшает производительность веб-приложения на устройствах пользователей за счет отсутствия JavaScript кода.

В различных реализациях «server-side rendering» ключевой концепцией является способность запускать JavaScript на сервере с целью создания веб-разметки. Далее эта сгенерированная разметка передается в браузер пользователю, и там начинается процесс построения DOM-дерева. Это позволяет отображать веб-страницы согласно типичному для веб-браузера сценарию. Таким образом, можно выделить три основных этапа в использовании подхода серверного рендеринга для одностраничных веб-приложений. Первый этап - запуск JavaScript-кода на сервере и создание статических HTML-страниц. Второй этап - оптимальная передача сгенерированной разметки клиенту. Третий этап - выполнение полученного кода в веб-браузере и построение DOM-дерева.

Для реализации этого подхода необходимо наличие платформы Node.js вне зависимости от выбранного фронтенд-фреймворка, соответственно, нужен сервер для осуществления рендеринга, а также необходимо предусмотреть способы по сериализации и десериализации разметки на клиенте и сервере.

Отмечая преимущества такого решения в виде увеличения производительности клиентской части приложения и улучшении SEO-оптимизации сайта за счет сгенерированных HTML-страниц, а не JavaScript кода, возникают и недостатки с точки зрения повышения расходов на поддержу приложения относительно расходов сервера, разработка более сложной архитектуры, что приводит к бóльшим затратам на поддержку такого решения.

В работе «Optimize along the way: An industrial case study on web performance» авторы отмечают, что повышение производительности веб-приложений является комплексной задачей, поскольку требует глубокого понимания как механизма браузера, так и конкретных сценариев использования рассматриваемого веб-приложения. [7] За счет улучшения времени загрузки, производительность, воспринимаемая пользователем, увеличивается.

Различия в процессе рендеринга с использованием vDOM объясняются архитектурой фреймворков. Nian Li и Bo Zhang в статье «The Research on Single Page Application Front-end development Based on Vue» исследуют одностраничные приложения в рамках реализации на основе фреймворка Vue, отмечая следующий паттерн проектирования как ведущий в данном инструменте – MVVM [8].

Vue придерживается идее управления данными и компонентами, используя дизайн инкрементальной разработки. С паттерном MVVM данные (model) и представления (view) разделены таким образом, чтобы исключить прямое взаимодействие.

Для декомпозиции взаимодействия между двумя слоями приложения используется view-model, при помощи которого возможно отслеживание действий с обеих сторон и своевременное выполнение соответствующей операции реагирования на изменения, обновления данных и связывания элементов.

Схема архитектуры MVVM в приложении, реализованном с использованием фреймворка Vue, представлена на Рисунке 3.

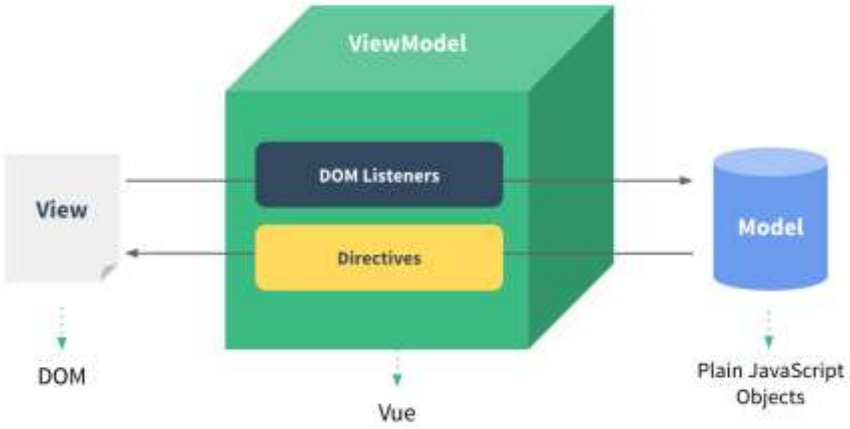

Рисунок 3 – Архитектура MVVM в приложении Vue

Принцип работы представляет собой двунаправленное связывание, после создания привязки, происходит синхронизация DOM с данными. В момент обновления данных, узлы DOM также будут синхронизироваться с данными. Если представление обновится в DOM, то view-model, в свою очередь, вызовет определенную логику приложения для запуска механизма обновления данных в модели (model), чтобы реализовать двунаправленное связывание.

В статье «Modern Web Frameworks: A Comparison of Rendering Performance» Risto Ollila, Niko Makitalo и Tommi Mikkonen рассмотрели, как работают стратегии рендеринга современных фронтенд-фреймворков и представили способ их относительной производительности. [10]

Таблица 2 – Описание факторов, влияющих на производительность исследуемых фреймворков

Фреймворк / Библиотека	**Обрабатываемые компоненты**	**Обработанные элементы**	**Наличие vDOM**
Vue	Только «грязные» компоненты	Только привязанные	Есть
Angular	Все	Только привязанные	Нет
React	Поддерево обновляемого компонента	Все	Есть
Svelte	Только «грязные» компоненты	Только привязанные	Нет

Таким образом, Vue и Svelte имеют преимущество, когда обновляются только небольшие части интерфейса, так как они ограничиваются обновлением только необходимых элементов.

В то время как Angular и React могут потреблять больше ресурсов на обновление элементов, даже если они не привязаны к изменяющимся данным.

Заключение

В процессе анализа было выявлено, что темы, связанные с процессами пререндеринга и регидратации (изоморфного рендеринга), остаются недостаточно исследованными и подробно освещены как в отечественных, так и в зарубежных исследованиях.

Анализ источников показал, что актуальность проблемы оптимизации рендеринга веб-приложений обусловлена динамичным развитием современных веб-приложений. По мере того, как приложения становятся все более сложными и функциональными, нагрузка на клиентскую сторону с использованием CSR (client-side rendering) значительно увеличивается, что отмечается в исследованиях и зарубежных, и отечественных авторов.

Таким образом, изучение процесса рендеринга и поиски оптимизационных решений в этой области являются крайне важными задачами для современных разработчиков и исследователей. Необходимо стремиться к нахождению баланса между богатством функциональности веб-приложений и их производительностью, чтобы обеспечить плавное и комфортное взаимодействие пользователей с приложениями в любых условиях.

Список литературы

1. Толстых, М. А. Оценка перспективности фреймворков для создания современных web-приложений//Научные исследования XXI века. – 2020. – №1(3). – С. 79–82.
2. Еремин М.В. Сравнительный анализ популярных JavaScript фронтенд решений//Тенденции развития науки и образования. – 2022. – №87– 1. – С. 64–68.
3. Заманов Е.А., Дутова Е.А., Селецкая Н.Г. Рендеринг и его влияние на архитектуру веб-приложений//Российский университет транспорта (Москва). – 2021. – С. 297–301.
4. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. Современные фреймворки для разработки web-приложений//Интеллектуальные технологии на транспорте. – 2020. – №4 (24). – С. 23–29.
5. Летон Г., Глазько П.Е. Исследование методологии оптимизации рендеринга компонентов на примере svelte//XI Конгресс Молодых Учёных Сборник научных трудов. Санкт-Петербург, 2022. – 2022. – С. 231– 234.
6. Ростов Д.С., Готская И.Б., Государев И.Б. Исследование методов имплементации рендеринга клиентского интерфейса на стороне сервера//XLVIII Научная и Учебно-Методическая Конференция Университета ИТМО Санкт-Петербург, 29 января – 01 февраля 2019 года. – 2019. – С. 244– 246.
7. Jasper van Riet, Ivano Malavolta, Taher A. Ghaleb Optimize along the way: An industrial case study on web performance//Journal of Systems and Software. – 2023. Volume 198
8. Nian Li, Bo Zhang The Research on Single Page Application Front-end development Based on Vue//Journal of Physics: Conference Series. – 2021. Volume 1883
9. Ollila R., Mäkitalo N, Mikkonen T. Modern Web Frameworks: A Comparison of Rendering Performance//Journal of Web Engineering. – 2022. – №21(03). – С. 789–814.

References

1. Tolstykh, M. A. Evaluation of the prospects of frameworks for creating modern web applications // Scientific research of the XXI century. – 2020. – №1(3). – Pp. 79-82.
2. Eremin M.V. Comparative analysis of popular JavaScript frontend solutions // Trends in the development of science and education. – 2022. – №87– 1. – Pp. 64-68.
3. Zamanov E.A., Dutova E.A., Seletskaya N.G. Rendering and its influence on the architecture of web applications // Russian University of Transport (Moscow). - 2021. – pp. 297-301.
4. Baidybekov A.A., Gilvanov R.G., Molodkin I.A. Modern frameworks for web application development // Intelligent technologies in transport. – 2020. – №4 (24). – Pp. 23-29.
5. Leton G., Glazko P.E. A study of the methodology for optimizing component rendering on the example of svelte // XI Congress of Young Scientists Collection of scientific papers. St. Petersburg, 2022. – 2022. – pp. 231-234.
6. Rostov D.S., Gotskaya I.B., Gosudarev I.B. Research of methods of implementation of rendering of the client interface on the server side // XLVIII Scientific and Educational-Methodical Conference of ITMO University St. Petersburg, January 29 – February 01, 2019. - 2019. – pp. 244-246.
7. Jasper van Riet, Ivano Malavolta, Taher A. Ghaleb Optimize along the way: An industrial case study on web performance // Journal of Systems and Software. – 2023. Volume 198
8. Nian Li, Bo Zhang The Research on Single Page Application Front-end development Based on Vue // Journal of Physics: Conference Series. – 2021. Volume 1883
9. Ollila R., Mäkitalo N, Mikkonen T. Modern Web Frameworks: A Comparison of Rendering Performance // Journal of Web Engineering. – 2022. – №21(03). – pp. 789-814