Волошин Александр Сергеевич
студент
Шурбаев Никита Александрович
студент

ФГБОУ ВО «Кубанский государственный аграрный
университет им. И.Т. Трубилина»
г. Краснодар, Краснодарский край

ВЫБОР FRONTEND-АРХИТЕКТУРЫ ДЛЯ ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ И ПОЛЬЗОВАТЕЛЬСКОГО ОПЫТА ВЕБ-ПРИЛОЖЕНИЙ

Аннотация: *авторы статьи отмечают, что в условиях стремительного роста требований к веб-приложениям, а именно скорости и интерактивности, ключевую роль играет выбор архитектуры. В статье рассматриваются вопросы эволюции SPA и MPA, значения метрик, положительных и отрицательных сторон самых популярных гибридных моделей и баланса между скоростью загрузки, интерактивностью и SEO. Цели исследования включают: анализ развития frontend-архитектур, их влияние на метрики Core Web Vitals и Search Engine Optimization, сравнение SSR, SSG и ISR.*

Ключевые слова: *frontend, SPA, MPA, SSG, SSR, ISR, CWV, SEO, производительность, веб-приложения, архитектура приложений.*

С развитием цифровизации объем интерфейсов в веб-приложениях значительно вырос, количество логики в клиентской части увеличилось, а требования пользователей к скорости работы и удобству взаимодействия стали жестче. Проблема начала ярко проявляться в середине 2010-х гг., когда SPA-архитектура (Single-Page Application) показала предел масштабируемости.

Для общего понимания качества пользовательского опыта на сайте, компанией Google была разработана система CWV (Core Web Vitals). Она представляет собой три основные метрики, которые дают понять, насколько хорошо оптимизирован сайт.

1. LCP (Longest Contentful Paint) – метрика, которая показывает скорость загрузки самого крупного элемента на странице. После недавнего обновления расчета LCP, метрика стала точнее. Это сделано благодаря приоритету загрузки элементов. Теперь первыми загружаются изображения, а потом происходит рендеринг страницы. Для PNG и GIF загрузка происходит в момент первого кадра, что сильно экономит время и делает оптимизацию лучше. Малодетализированные изображения (например, фон) не включаются в расчет, то есть скорость их загрузки не измеряется метрикой, что позволяет ей быть более точной.

Ухудшить LCP может медленный сервер, слишком громоздкий код или отсутствие кеша. Это часто заставляет разработчика внедрять серверный рендеринг, оптимизировать запросы, использовать CDN (Content Delivery Network).

2. INP (Interaction to Next Paint) – это метрика измеряет скорость реакции сайта на клик. Главное отличие этой метрики от FID (First Input Delay), который был в применении до 2024 г., в том, что он оценивает время задержки на протяжении всей сессии, а не при первом контакте пользователя с сайтом, как это было с FID. В том числе метрика теперь принимает во внимание задержку при нажатии клавиш и фиксирует взаимодействия с контекстным меню.

Замедлять отклик может большое количество анимаций или сторонние скрипты.

3. CLS (Cumulative Layout Shift) – метрика, отвечающая за сдвиг элементов на странице, что приводит к нежелательным кликам и затруднительному чтению контента. Google улучшил расчет CLS, игнорируя сдвиги макета во время перемещения по странице и выделения или перетаскивания элементов мышью. Это особенно важно для интерактивных сайтов. действие помогло снизить вероятность ложных срабатываний и позволило избежать необоснованно высоких показателей CLS. Благодаря загрузке изображений перед рендерингом страницы не происходит сдвигов контента из-за поздней загрузки графики [1].

Несмотря на то, что CWV являются важными факторами ранжирования, содержимое веб-страницы остается самым важным компонентом. Так же релевантность поиска и ссылочный профиль важнее чем CWV.

Страницы, имеющие наиболее высокое ранжирование, соблюдают тенденцию низкого LCP. Рекомендуемое значение LCP до 2,5 с. включительно, только 39% страниц соблюдают его [2].

Официальные представители Google заявляют, что релевантность остается самым важным фактором ранжирования. Например, Д. Мюллер говорил, что Core Web Vitals могут вносить небольшие изменения в ранжирование, в то время как другие факторы уже оптимизированы [3].

Именно невозможность SPA-архитектуры обеспечить приемлемые значения метрик LCP, FID и CLS стала причиной перехода на следующий этап развития веб-приложений, а именно использование MPA-архитектуры (Multi-Page Application), так как именно многостраничная парадигма давала то, что перестали обеспечивать SPA при росте сложности.

1. MPA изначально визуализировал содержимое страницы на сервере, что давало предсказуемое время первого отображения и минимальную нагрузку на браузер.

2. HTML, сгенерированный статически или на сервере, индексируется поисковыми системами легче, чем динамически формируемый, как в случае SPA, что улучшает SEO (Search Engine Optimization) [4; 5].

3. MPA намного эффективнее на слабых устройствах, так как минимизирует вычислительные нагрузки на стороне клиента [6].

Несмотря на преимущества MPA в сравнении SPA, многостраничные приложения тоже имели свои ограничения, которые сделали дальнейшую эволюцию архитектур неизбежной. Каждое действие пользователя требовало полной повторной отрисовки страницы, что в свою очередь сильно увеличивало нагрузку на сервер и препятствовало масштабируемости веб-страницы при росте числа пользователей. Множественные перезагрузки страницы приводили к снижению непрерывности пользовательского опыта и усложняли работу с динамическими данными [4].

Реализация сложной клиентской логики, анимаций и локальных обновлений были усложнены, потому что повторная загрузка увеличивала сетевой трафик и замедляла отклик интерфейса.

Параллельно с эволюцией frontend-архитектур происходило развитие языка программирования JavaScript и его фреймворков, что напрямую влияло на возможности SPA, MPA и последующих гибридных моделей. Появление таких фреймворков, как Angular, React и Vue позволило создавать намного более сложные и интерфейсы, реализовывая логику на клиентской части. Использование AJAX (Asynchronous JavaScript and eXtensible Markup Language) и других технологий для асинхронного обмена данными позволило динамически обновлять части страницы без полной перезагрузки [7].

Следующим этапом развития frontend-архитектур стало появление гибридных моделей, которые объединили в себе быструю начальную отрисовку, предсказуемость отображения содержимого (от MPA), а также высокую интерактивность интерфейса и возможность частичного обновления отдельных компонентов вместо полной перезагрузки веб-приложения (от SPA).

Основными гибридными архитектурами стали следующие.

1. SSR (Server-Side Rendering). Подход генерации страницы, при котором HTML (HyperText Markup Language) формируется на сервере в момент запроса со стороны клиента. Он улучшил показатели метрики FCP и усовершенствовал индексируемость страниц поисковыми системами, но при этом при большом количестве запросов значительно повысил нагрузку на сервер.

2. SSG (Static Site Generation). Подход, при котором HTML создается на этапе сборки проекта, а не динамически. Он обеспечил высокую производительность на всех устройствах, полную совместимость с поисковыми системами и простоту масштабируемости веб-страницы, но при такой архитектуре возможна сложность интеграции с системами, в которых данные обновляются в реальном времени.

3. ISR (Incremental Static Regeneration). Подход, объединяющий преимущества SSG и возможность обновления содержимого веб-страницы в реальном времени без полной пересборки. Этот метод сохранил преимущества статической генерации, уменьшив нагрузку на серверную часть, но он сложнее в настройке,

так как требует дополнительной инфраструктуры для отслеживания изменений и не подходит для мгновенно обновляемых данных [8; 9].

Из-за различий подходов к генерации HTML у разных гибридных архитектур веб-приложений, ключевые метрики CWV и SEO, влияющие на пользовательский опыт, тоже разнятся в значениях. Выбор конкретной архитектуры зависит от требований к производительности, планируемой интерактивности интерфейса и особенностей контента [10].

SSR оптимален для веб-приложений с динамическим контентом, в которых важна актуальность данных. Повышенная нагрузка на сервер при увеличении количества запросов компенсируется быстрой начальной отрисовкой и высокой индексируемостью поисковыми системами. Выбор этого архитектурного решения оправдан в случаях, когда важна высокая интерактивность интерфейсов и маленькое время отклика при большом количестве пользовательских запросов

SSG за счет предварительной генерации страниц обеспечивает стабильность макета и высокую скорость загрузки. Такой подход улучшает показатели метрик LCP и CLS. Основным ограничением является статический характер данных. Такой выбор рационален в случае разработки веб-приложений, не требующих мгновенных обновлений и ориентированных на статическое содержимое.

ISR объединяет в себе преимущества SSG и возможность обновления отдельных компонентов веб-страницы при сохранении высокой производительности. Эта архитектура нагружает серверную часть значительно меньше по сравнению с SSR и позволяет поддерживать достаточно низкие показатели LCP и CLS. Данное решение уместно в проектах, содержащих как статические, так и динамические блоки, то есть важны высокая скорость, стабильность интерфейса и актуальность данных (LCP, CLS и INP) [5; 10].

Таким образом, невозможно выделить универсальное архитектурное решение, так как каждый метод имеет как сильные, так и слабые стороны. SSR обеспечивает быструю начальную отрисовку и способен обеспечить актуальность данных, что делает его предпочтительным для интерактивных интерфейсов. SSG гарантирует стабильность интерфейса и высокую производительность, но при

этом не поддерживает мгновенные обновления, что значительно сужает область его применения. ISR, несмотря на сочетание преимуществ SSG и возможности обновления отдельных компонентов, требует значительно более структурно сложной инфраструктурной поддержки.

Для веб-приложений, в которых важны высокая динамичность и интерактивность интерфейса стоит применять SSR (например, новостные порталы или интернет-магазины).

Для сайтов с преимущественно статическим содержимым целесообразным выбором будет SSG (например, информационных или корпоративных сайтов).

Для проектов смешанного типа, где есть и статические, и динамические блоки оптимален ISR (например, интернет-магазины с блоками каталогов и акций).

Тем не менее, при ускоряющемся развитии информационных технологий, повсеместной цифровизацией следует учитывать существующие угрозы коммуникационным потокам, экономическим операциям и другим сферам [11].

Список литературы

1. Stubbs A. A Pocket Guide to Core Web Vitals: Optimising User Experience for the Modern Web (Pocket Guides To Web Development) / A. Stubbs. – Independently published, 2023. – 125 p.

2. Влияние Core Web Vitals на бизнес [Electronic resource]. – Access mode: https://web.dev/case-studies/vitals-business-impact?hl=ru (date of application: 02.12.2025).

3. Сравнение одностраничных приложений (SPA) с многостраничными (MPA) // SCAND [Электронный ресурс]. – Режим доступа: https://scand.com/ru/company/blog/single-page-application-vs-multi-page-application/ (дата обращения: 02.12.2025).

4. Matthew E. Tech SEO Guide: A Reference Guide for Developers and Marketers Involved in Technical SEO / E. Matthew. – Apress, 2023. – 164 p.

5. Основы SEO и современные вызовы // SberUniversity [Электронный ресурс]. – Режим доступа: https://developers.sber.ru/help/business-development/seo-optimization#sovremennye-vyzovy-vandnbspseo-optimizacii (дата обращения: 02.12.2025).

6. Richards M. Fundamentals of Software Architecture / M. Richards, N. Ford. – O'Reilly, 2020. – 23 p.

7. Чиннатхамби К. Изучаем React / К. Чиннатхамби; пер. с англ. – М.: Бомбора, 2019. – 368 с.

8. Incremental Static Regeneration (ISR) и Edge Rendering: современный подход к скорости и SEO-оптимизации фронтенда // YLab University [Электронный ресурс]. – Режим доступа: https://university.ylab.io/articles/tpost/6dis8h35o1-incremental-static-regeneration-isr-i-ed (дата обращения: 02.12.2025).

9. Server Side Rendering vs Static Site Generation vs Incremental Static Regeneration [Electronic resource]. – Access mode: https://www.react-bricks.com/blog/server-side-rendering-vs-static-site-generation-vs-incremental-static-regeneration (date of application: 02.12.2025).

10. Эспозито Д. Разработка современных веб-приложений. Анализ предметных областей и технологий / Д. Эспозито; пер. с англ. – М.: Вильямс. – 464 с.

11. Российские информационные вызовы и ответы на них / Г.В. Арустамян, Д.О. Шестак, Р.А. Дилбандян [и др.] // Культура Мира. – 2025. – Т. 13. №47(4). – С. 181–195. EDN IKDNZR

12. Российские ответы на технологические вызовы: стратегии и решения в условиях современного менеджмента / О.В. Терещенко, Д.В. Еськов, С.А. Шульга [и др.] // Естественно-гуманитарные исследования. – 2025. – №4(60). – С. 842–847. EDN PFQHPU