Научная статья
УДК 004.413
doi: 10.17586/2713-1874-2022-1-41-47

КРОССПЛАТФОРМЕННАЯ РАЗРАБОТКА НА БАЗЕ ВЕБ-ТЕХНОЛОГИЙ ДЛЯ ПОДДЕРЖКИ РЕШЕНИЙ В ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ СИСТЕМАХ УПРАВЛЕНИЯ

Александр Юрьевич Пчелкин[1]✉, Наталия Федоровна Гусарова[2]

[1,2]Университет ИТМО, Санкт-Петербург, Россия
[1]sapchelkin@yandex.ru✉
[2]nfgusarova@itmo.ru, https://orcid.org/0000-0002-1361-6037
Язык статьи – русский

Аннотация: На фоне высокой конкуренции и необходимости адаптироваться к постоянно изменяющимся условиям и потребностям рынка многие компании начинают отдавать предпочтение технологиям, позволяющим повысить гибкость и скорость разработки, а также снизить количество привлекаемых к этой разработке специалистов. В статье рассматриваются веб-технологии и подходы к их применению, позволяющие преобразовывать веб-приложения в мобильные и настольные, расширяя их функционал и возможности для решения задач управления в проблемно-ориентированных системах, описаны результаты применения технологий для медицинских приложений.

Ключевые слова: веб-технологии, коллективная разработка, кроссплатформенная разработка, проблемно-ориентированные системы, прогрессивные веб-приложения, системы поддержки принятия решений

Ссылка для цитирования: Пчелкин А.Ю., Гусарова Н.Ф. Кроссплатформенная разработка на базе веб-технологий для поддержки решений в проблемно-ориентированных системах управления // Экономика. Право. Инновации. 2022. № 1. С. 41–47. http://dx.doi.org/10.17586/2713-1874-2022-1-41-47.

CROSS-PLATFORM DEVELOPMENT BASED ON WEB TECHNOLOGIES TO SUPPORT SOLUTIONS IN PROBLEM-ORIENTED MANAGEMENT SYSTEMS

Alexander Yu. Pchelkin[1]✉, Natalia F. Gusarova[2]

[1,2]ITMO University, Saint Petersburg, Russia
[1]sapchelkin@yandex.ru✉
[2]nfgusarova@itmo.ru, https://orcid.org/0000-0002-1361-6037
Article in Russian

Abstract: Against the background of high competition and the need to adapt to constantly changing conditions and market needs, many companies are beginning to prefer technologies that allow for increased flexibility and speed of development, as well as reduce the number of specialists involved in this development. The article discusses web technologies and approaches to their application, allowing to transform web applications into mobile and desktop, expanding their functionality and capabilities to solve management problems in problem-oriented systems, describes the results of the use of technologies for media applications.

Keywords: web technologies, collective development, cross-platform development, problem-oriented systems, progressive web applications, decision support systems

For citation: PchelkinA.Yu., Gusarova N.F. Cross-Platform Development Based on Web Technologies to Support Solutions in Problem-Oriented Management Systems. *Ekonomika. Pravo. Innovacii.* 2022. No. 1. pp. 41–47. (In Russ.). http://dx.doi.org/10.17586/2713-1874-2022-1-41-47.

Введение. Функционал современных веб-приложений практически не уступает настольным и мобильным версиям [1]. При этом разработанное веб-приложение, за некоторыми оговорками, будет запускаться и работать у пользователей независимо от типа их устройства или операционной системы. Если перед компанией стоит задача при разработке приложения покрыть максимально широкую аудиторию, то разработка нативных версий под каждое устройство займет огромное количество времени и потребует

привлечения большого числа специалистов разных направлений. При этом есть риск того, что скорость этой разработки окажется крайне невысокой из-за необходимости постоянной синхронизации разных версий. Если же к этому прибавить еще и высокую конкуренцию в IT-индустрии и скорость развития технологий, то помимо серьезных финансовых потерь подобный подход может привести и к потере конкурентоспособности, и актуальности разработки. В тоже время существует большое число пользователей, которые предпочитают пользоваться именно нативными версиями приложений. Также статистика показывает, что хотя с привлечением новых пользователей лучше справляются веб-приложения, процент удержания у нативных версий выше.

Помимо этого, нативные версии имеют ряд других преимуществ. Зачастую они могут работать без интернета, имеют некоторые возможности, недоступные веб-версиям, работают быстрее, и их можно продвигать в магазинах приложений устройств.

В данной статье рассматриваются веб-технологии, такие как ServiceWorker, PWA, ReactNative и Electron, позволяющие расширять возможности веб-приложений, а также собирать на их основе приложения, которые пользователь сможет установить на свое устройство [2].

Цель исследования. Цель работы – исследовать существующие популярные веб-технологии кроссплатформенной разработки, определить границы их возможностей и применения.

Методы и материалы исследования. Для исследования были отобраны актуальные популярные фреймворки и технологии кроссплатформенной разработки. Ключевым фактором при отборе было практически полное сохранение основной кодовой базы, вне зависимости от платформы. То есть выбранная технология должна была дополнять изначальное веб-приложение, расширяя его возможности и функционал без серьезных изменений в основном коде.

Каждая технология рассматривалась с точки зрения того, какие возможности она предоставляет разработчику и насколько трудоемко их внедрение. Для этого изучалась официальная документация, открытые репозитории проектов, посвященные разработке статьи, рассматривались примеры существующих приложений. После этого каждая выбранная технология в рамках нескольких различных проектов была применена на практике.

Основная часть. Веб-технологии активно развиваются, открывая новые возможности для разработчиков. Одни возможности позволяют расширить функционал, другие упростить и удешевить разработку. Эти технологии оказываются востребованными как в стартапах, где крайне важна скорость реализации идей, так и в крупных компаниях, где под тяжестью масштабов разработки, параллельное развитие нескольких версий становится неподъемной задачей.

Нативные приложения все еще зачастую остаются лучшим решением при разработке под какую-то конкретную платформу. Однако при кроссплатформенной разработке, разница между приложением на базе вебтехнологий и нативным приложением оказывается настолько незначительной [3] на фоне различий в необходимых трудозатратах, что многие компании полностью отказываются от нативных версий.

При сравнении веб-приложений и нативных, в качестве достоинств последних, приводят следующие факторы:

– приложение может работать без интернета;

– более высокая производительность;

– возможность задействовать функции устройства, такие как камера, геолокация и адресная книга;

– возможность публикации в магазине приложений.

Однако все это реализуемо и в рамках веб-технологий.

Первой рассмотренной технологий стал **ServiceWorker**. ServiceWorker – это скрипт, который запускается браузером в фоновом режиме, отдельно от страницы. При помощи технологии ServiceWorker можно решать разные задачи, но самое важное – это возможность использовать его в качестве посредника между веб-приложением и сетью Интернет (Рисунок 1). Тогда при помощи данного скрипта можно перехватывать все исходящие из приложения запросы и обрабатывать их.

Страница веб-приложения, которую пользователь видит в своем браузере, пред-

ставляет собой набор статических ресурсов, включающих в себя каркас приложения, стили, скрипты, шрифты и т.д. Все эти данные можно при помощи ServiceWorker кэшировать в браузере пользователя, после чего, при их повторном запросе, обращаться не к серверу, а извлекать и возвращать данные из внутреннего хранилища.

Таким образом, все необходимые для запуска приложения ресурсы будут уже находиться на устройстве, что позволит пользователю открывать веб-приложение, даже если подключение к интернету отсутствует. Помимо статических ресурсов, при помощи ServiceWorker можно различными способами сохранять и другие данные.

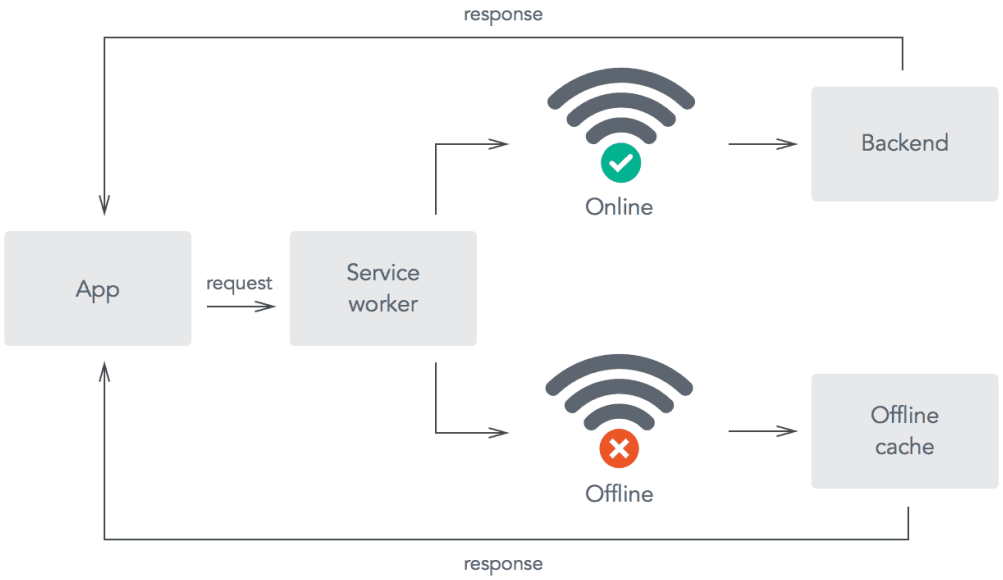

Рисунок 1 – Схема работы ServiceWorker

Существует множество стратегий кеширования. Среди наиболее популярных можно выделить cacheFirst и networkFirst.

Первая применяется для кэширования неизменяющихся со временем данных. Т.е. если каждый раз при переходе по ссылке сервер отдает одни и те же данные, то один раз сохранив их в кэш, к серверу можно больше не обращаться.

Если же данные меняются, то можно использовать вторую стратегию. Тогда данные будут запрашиваться с сервера каждый раз, но в случае, если сервер не отвечает или интернет отсутствует, будут возвращены результаты последнего удачного запроса.

Помимо этого, можно кэшировать не только данные, к которым пользователь обращался, пока был онлайн. К примеру, некоторые новостные сайты могут подгружать актуальные новости в фоновом режиме, благодаря чему пользователь может получать новую для него информацию, даже когда будет вне сети.

Также, несмотря на то, что без интернета пользователь не сможет взаимодействовать с сервером, это не означает, что необходимо ограничиться только операциями на чтение данных. Можно организовать фиксацию всех изменений, производимых пользователем офлайн, после чего синхронизировать их с сервером при восстановлении соединения.

Таким образом, в рамках веб-приложения можно обеспечить полный доступ к функционалу системы пользователю, даже если у того отсутствует подключение к интернету.

Технология ServiceWorker поддерживается практически любым современным браузером. По статистике свыше 96% пользователей используют для выхода в сеть браузеры, имеющие полную поддержку Service Worker.

Если к веб-приложению, помимо ServiceWorker, добавить еще несколько технологий, то приложение можно сделать устанавливаемым. Такие приложения называются прогрессивными (**PWA – ProgressiveWebApplication**) [4].

Помимо возможности работы офлайн и запуска в виде отдельного приложения с иконки рабочего стола, такие приложения позволяют отправлять пользователю push-

уведомления и имеют более широкий доступ к аппаратному обеспечению устройства.

Устанавливать такое приложение можно как на смартфон, так и на персональный компьютер. При этом нет необходимости отправлять пользователя в магазин приложений устройства. Установку можно совершить прямо со страницы веб-приложения. Для этого не требуются права администратора, и осуществляется вопреки запрету устанавливать приложения из неизвестных источников. Также, так как приложение устанавливается не через магазин устройства, то и нет необходимости отдавать процент с прибыли.

Еще одним достоинством прогрессивных приложений является легкость их обновлений. Несмотря на то, что вся необходимая для работы приложения статика кэшируется на устройстве пользователя и больше не запрашивается с сервера, изменения ServiceWorker все еще отслеживаются. При выпуске обновления, разработчику достаточно внести информацию о новой версии в файл ServiceWorker, и PWA на устройстве пользователя автоматически произведет его переустановку при следующем запуске.

PWA активно продвигаются компанией Google. Поэтому такие приложения можно публиковать в их магазине приложений GooglePlay. Также их можно публиковать в магазинах Microsoft Store и Samsung Galaxy Store. В магазине AppleAppStore публиковать такие приложения не разрешено.

Из ограничений стоит еще выделить, что возможность прямой установки тоже доступна не из всех браузеров. Однако по статистике 86% пользователей используют браузеры, поддерживающие эту функцию.

PWA, по сути, является просто надстройкой. Благодаря этому даже если какие-то из вышеописанных технологий браузером пользователя не поддерживаются, он все равно сможет открыть сайт и работать с ним, как с обычным веб-приложением.

На данный момент многие крупные компании начинают использовать PWA. Среди них Google и AliExpress.

Если возможностей PWA оказывается недостаточно для реализации идеи, то окончательно стереть грань между нативным и веб-приложением можно, используя **React Native** или **Electron.** При этом даже не обязательно использовать их в том виде, в котором изначально задумывалось их создателями.

К примеру, React Native – это кроссплатформенный фреймворк для разработки нативных приложений. При разработке используются универсальные компоненты самого фреймворка, которые при сборке приложения под конкурентную платформу заменяются интерфейсными компонентами выбранной платформы [5]. Т.е. несмотря на то, что разработка приложения осуществляется на языке JavaScript, базируясь на принципах работы React, конечным результатом является нативное приложение.

Если разработка проекта начинается с нуля, то React Native будет отличным кроссплатформенным решением. Однако если уже есть разработанная веб-версия, то для создания нативного приложения потребуется существенно переработать имеющийся код, например, переписать существующие элементы интерфейса, используя компоненты React Native. В этом случае можно сделать приложение гибридным, задействовав компонент WebView из React Native [6]. WebView – это компонент, который позволяет встраивать веб-страницы в приложения, своеобразный минибраузер.

Идея такой реализации близка к разработке на ApacheCordova (PhoneGap) [7], где написанный на веб-технологиях пользовательский интерфейс отображался через средство просмотра веб-страниц, а функционал приложения расширялся за счет подключения специальных плагинов.

В React Native существуют механизмы для вызова функций и обмена сообщениями между основным приложением и контентом, отображаемым во WebView. Таким образом становится доступным взаимодействие с устройством пользователя, выходящее за рамки обычных возможностей веб-приложения. К примеру, доступ к камере, геолокации или адресной книге.

Разница между гибридным приложением и нативным – в скорости работы пользовательского интерфейса. Однако даже на слабом устройстве пользователь не заметит между ними разницы, если должным образом проведены работы по оптимизации (не происходит лишних перерисовок компонентов, результаты выполнения сложных вычисле-

ний и часто вызываемые элементы интерфейса мемоизируются, не создаются лишние обработчики событий) [8].

Также разработчик может в любых пропорциях совмещать нативные компоненты и контент, отображаемый через WebView. К примеру, сделав элементы навигации и некоторые страницы полностью нативными, а другие – отображаемыми через средство просмотра веб-страниц. Вне зависимости от того, является ли приложение нативным или гибридным, для смартфона оно будет полноценным мобильным приложением. Такое приложение можно будет свободно публиковать в магазине приложений не только устройств под управлением операционной системы Android, но и iOS.

Для разработки настольных приложений можно использовать Electron. Electron – это фреймворк, разработанный GitHub, в основе которого лежат Chromium и Node.js, объединённые в единую среду. Т.е. получается, что это некое клиент-серверное приложение, собранное вместе. Пользовательский интерфейс отображается через веб-браузер Chromium, а сервер на Node.js обрабатывает его запросы. И хотя такая конструкция выглядит достаточно странной, она хорошо работает и предоставляет разработчику широкие возможности. Помимо возможности использовать в проекте любые JavaScript библиотеки, к приложению на Electron можно подключать модули, написанные на других языках, вроде Python или C++. Благодаря этому в разрабатываемом приложении можно задействовать уже их библиотеки или писать оптимизированный код для выполнения сложных вычислений.

Есть у подобного подхода и недостатки. Так как приложение базируется на Chromium, то для работы ему будет требоваться немного больше оперативной памяти, чем аналогичному нативному приложению. Так же Chromium включается в итоговую сборку проекта, поэтому размер установочного файла будет не менее 30Мб.

Electron активно используется в разработке. К примеру, на нем написаны настольные версии Skype, Discord, Slack, VisualStudioCode и Atom.

Полученные результаты. Рассмотренные выше технологии были успешно апробированы при разработке нескольких собственных проектов. Так использование технологии Apache Cordova при создании мобильного медицинского сервиса для самоорганизации пожилых людей во время пандемии [9] позволило добавить в приложение функционал, задействующий камеру и микрофон устройства. При дальнейшем переходе приложения на более актуальную и развивающуюся технологию React Native удалось практически полностью сохранить исходную кодовую базу.

Технологии ServiceWorker и PWA были применены при разработке краудсорсинговой платформы для сбора и разметки медицинских снимков. Это позволило, во-первых, реализовать off-line режим работы web-приложения и, во-вторых, сделать его устанавливаемым на устройство пользователя (Рисунок 2). Эти возможности были крайне востребованы, так как предполагалось использование приложения в регионах с низким качеством интернет связи.

Рисунок 2 – Краудсорсинговая платформа на разных устройствах

Краудсорсинговая платформа реализует концепцию Human-in-the-Loop, при которой человек интегрирован в обучение искусственного интеллекта. Специалист итеративно формирует обучающие данные и контролирует процесс обучения, в то время как искусственный интеллект становится все более точным и в какой-то момент уже сам начинает оказывать помощь специалисту в принятии решений в изучаемой области (Рисунок 3).

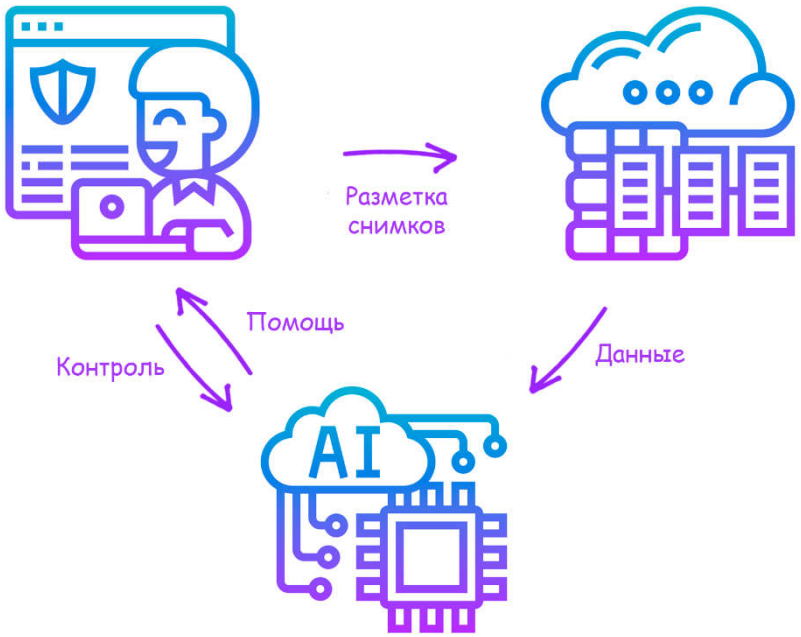

Рисунок 3 – Концепция Human-in-the-Loop

Технология Electron, позволяющая задействовать в приложении модули других языков (в том числе и Python), открывает возможности прямой интеграции результатов обучения искусственного интеллекта в настольную версию системы.

Вывод. Веб-технологии можно использовать не только при разработке сайтов и веб-приложений, но и при создании мобильных и настольных приложений. Их использование позволяет упростить и удешевить кроссплатформенную разработку, чем активно пользуются как крупные компании, так и небольшие стартапы.

В зависимости от потребностей можно использовать те или иные описанные выше технологии. При помощи ServiceWorker можно обеспечить офлайн доступ к приложению. Добавив к ServiceWorker несколько других технологий, можно сделать веб-приложение устанавливаемым. Если необходимо сделать полноценное кроссплатформенное приложение с полным доступом к функциям устройства и возможностью публикации в магазине приложений любой операционной системы, то можно рассмотреть вариант разработки нативного или гибридного приложения на ReactNative. При разработке настольных версий в качестве альтернативы можно рассмотреть фреймворк Electron, который позволяет еще сильнее расширить функционал приложения, благодаря возможности интеграции в него модулей, написанных на других языках.

Список источников

1. Шинкарев А.А. Ретроспектива развития веб-технологий в создании корпоративных информационных систем // Вестник Южно-Уральского государственного университета. Серия: Компьютерныетехнологии, управление, радиоэлектроника. 2020. 14 с.
2. Scott Adam D. JavaScript Everywhere: Building

References

1. Shinkarev A.A. A Retrospective of The Development of Web Technologies In The Creation of Corporate Information Systems.*VestnikYUzhno-Ural'skogogosudarstvennogouniversiteta. Seriya: Komp'yuternyetekhnologii, upravlenie, radioelektronika.* 2020. 14 p. (In Russ.).
2. Scott Adam D. JavaScript Everywhere: Building

Cross-Platform Applications with GraphQL, React, React Native, and Electron // O'Reilly Media. 2021. (In Eng.).
3. Barros L.P., Medeiros F., Moraes E.C., Feitosa A. Jnr. Analyzing the Performance of Apps Developed by using Cross-Platform and Native Technologies // Conference: International Conference on Software Engineering and Knowledge Engineering. 2020. (In Eng.).
4. Bhatt K. Progressive Web Application-Present and Future. 2019 [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/ publication/337544344_Progressive_Web_Applica tion-Present_and_Future (In Eng.).
5. Недяк А.В., Рудзейт О.Ю., Зайнетдинов А.Р., Рагулин П.Г. Инструменты мобильной кросс-платформенной разработки приложений // Интернет-журнал «Отходы и ресурсы». 2020. № 4.
6. Adinugroho T.Y., Gautama J.B. Review of Multi-platform Mobile Application Development Using WebView: Learning Management System on Mobile Platform // Procedia Computer Science. 2015. Т. 59. С. 291 (In Eng.).
7. Appiah F., Hayfron-Acquah J.B., Panford J.K., Twum F. A Tool Selection Framework for Cross Platform Mobile App Development // International Journal of Computer Applications. 2015. Т. 123. №. 2. 14 с. (In Eng.).
8. Rieger C., Majchrzak T.A. Towards the Definitive Evaluation Framework for Cross-Platform App Development Approaches // Journal of Systems and Software. Т. 153. 2019. 175 с. (In Eng.).
9. Pchelkin A., Gusarova N., Dobrenko N., Vatyan A. Mobile Healthcare Service for Self-organization in Older Populations During a Pandemic // Communications in Computer and Information Science. 2021. С. 379–390. (In Eng.).

Cross-Platform Applications with GraphQL, React, React Native, and Electron. *O'Reilly Media.* 2021.

3. Barros L.P., Medeiros F., Moraes E.C., Feitosa A. Jnr. Analyzing the Performance of Apps Developed by using Cross-Platform and Native Technologies. *Conference: International Conference on Software Engineering and Knowledge Engineering.* 2020.

4. Bhatt K. Progressive Web Application-Present and Future. 2019. Available at: https://www.researchgate.net/publication/337544344_Progressive_Web_Application-Present_and_Future

5. Nedyak A.V., Rudzeyt O.U., Zainetdinov A.R., Ragulin P.G. Mobile Cross-platform App Development Tools. *Internet-zhurnal «Othody i resursy».* 2020. No. 4. (In Russ.).
6. Adinugroho T.Y., Gautama J.B. Review of Multi-platform Mobile Application Development Using WebView: Learning Management System on Mobile Platform. *Procedia Computer Science.* Vol. 59. 2015. p. 291.
7. Appiah F., Hayfron-Acquah J.B., Panford J.K., Twum F. A Tool Selection Framework for Cross Platform Mobile App Development. *International Journal of Computer Applications*. 2015. Vol. 123. No. 2. 14 p.
8. Rieger C., Majchrzak T.A. Towards the Definitive Evaluation Framework for Cross-Platform App Development Approaches. *Journal of Systems and Software.* Vol. 153. 2019. p. 175.
9. Pchelkin A., Gusarova N., Dobrenko N., Vatyan A. Mobile Healthcare Service for Self-organization in Older Populations During a Pandemic. *Communications in Computer and Information Science.* 2021. pp. 379–390.

Статья поступила в редакцию24.12.2021; одобрена после рецензирования 29.12.2021; принята к публикации 16.01.2022.
The article was submitted 24.12.2021; approved after reviewing 29.12.2021; accepted for publication 16.01.2022.