УДК 004.8

ОБЗОР ПОДХОДОВ К СОЗДАНИЮ КРОССПЛАТФОРМЕННЫХ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ С ПРИМЕНЕНИЕМ ВЕБ-ТЕХНОЛОГИЙ

Д. В. Сиромский

Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газеты «Красноярский рабочий», 31
E-mail: dvmmod@yandex.ru

Рассматриваются обзор подходов к созданию кроссплатформенных мобильных приложений. Приведены особенности рассматриваемых технологий, а также их сравнение и итог сравнения.

Ключевые слова: мобильные технологии, прогрессивное веб приложение, мобильное приложение, кроссплатформенные мобильные приложения.

OVERVIEW OF APPROACHES TO CREATING CROSS-PLATFORM MOBILE APPLICATIONS USING WEB TECHNOLOGIES

D. V. Siromskiy

Reshetnev Siberian State University of Science and Technology
31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation
E-mail: dvmmod@yandex.ru

This article provides an overview of approaches to building cross-platform mobile applications. The features of the technologies under consideration are given, as well as their comparison and the result of comparison.

Keywords: mobile technologies, progressive web application, mobile application, cross-platform mobile applications.

Развитие веб-технологий постепенно привело к возможности создания сложных систем с меньшими усилиями, закономерно возникла идея применить лаконичные веб-технологии и библиотеки для создания нативных приложений. В результате возникли такие инструменты как: *Cordova, React Native, NativeScirpt, Progressive Web Application.*

Apache Cordova – позволяет программистам создавать приложения для мобильных устройств с помощью *CSS3*, *HTML5* и *JavaScript*, вместо того, чтобы использовать конкретные платформы *API* [1].

Приложения созданные с применением данного инструмента не являются полноценными мобильными приложениями, так как генерация макета происходит с применением системного компонента отвечающего за открытие веб-страниц в рамках других приложений.

React – *JavaScript*-библиотека для создания пользовательских интерфейсов [2].

React Native – это кроссплатформенный фреймворк с открытым исходным кодом для разработки нативных мобильных и настольных приложений на *JavaScript* и *TypeScript*, созданный *Facebook, Inc*. *React Native* поддерживает такие платформы как *Android, Android TV, iOS, macOS, Apple tvOS, Web, Windows* и *UWP*, позволяя разработчикам использовать возможности библиотеки *React* вне браузера для создания нативных приложений, имеющих полный доступ к системным API платформ [3].

NativeScirpt – это фреймворк с открытым исходным кодом, разрабатываемый компанией *Telerik*, для разработки приложений на платформах *Android* и *iOS*. Приложения *NativeScript* разрабатываются на платформонезависимых языках, таких как *JavaScript* или *TypeScript*. В *NativeScript* реализована полная поддержка фреймворка *Angular* [4].

Мобильные приложения, построенные с *NativeScript*, имеют полный доступ к API платформы так, будто они были разработаны в *XCode* или в *Android Studio*. Также разработчики могут включать в свои приложения сторонние библиотеки с таких ресурсов, как *Cocoapods*, *Android Arsenal*, *Maven* и *npm.js*, без создания дополнительных прослоек.

Progressive Web Application – технология в *web*-разработке, которая визуально и функционально трансформирует сайт в приложение (мобильное приложение в браузере)[5].

Чтобы сделать из сайта *PWA*, необходимо добавить к нему:

Service Worker – это *JavaScript*-файл, который запускается в фоновом режиме как автономный сервис. Он не связан с *DOM* (*Document Object Model*) или *web*-страницами, работает на другом потоке и получает доступ к *DOM* с помощью *API postMessage*. С точки зрения пользователя *Service Worker* позволяет выполнять такие действия, как, например, отправка *push*-уведомлений и предварительная загрузка материалов для просмотра в автономном режиме офлайн.

Сравнительная таблица

Технология	Недостатки	Преимущества
Apache Cordova	Поддержка существующих плагинов. Особенности браузерных компонентов *web-view*. Неудобная работа с файловой системой. Многозвенная схема отладки приложения. Большой вес исполняемого файла	Доступ к функциям устройства. Плагины. Лёгкая интеграция с приложениями имеющими веб интерфейс. Разнообразие фреймворков для разработки одностраничных приложений
React Native	Сложно адаптировать под все устройства. Разметка на собственных компонентах	Доступ к функциям устройства. Плагины. Поддержка *TypeScript*
NativeScript	Своя система верстки. Разметка на собственных компонентах	Доступ к функциям устройства. Плагины. Поддержка *TypeScript, Angular* и *Vue*
Progressive Web Application	Ограниченная поддержка функций устройства. Некоторые функции устройства недоступны с использованием текущих возможностей веб-браузера. Браузер-зависимый. *PWA* хорошо работает на самых последних версиях популярных браузеров	Доступ к функциям устройства. Поддержка *SEO*. В отличие от нативных приложений, *PWA* имеют *URL*-адреса, поэтому их можно проиндексировать в *Google*. Занимают меньше места в памяти устройства

Application shell – это виртуальная оболочка. Подобно оболочке нативного приложения, она загружается при его запуске, а далее динамическая информация загружается на неё из сети.

Web App Manifest предоставляет информацию о приложении в текстовом JSON-файле. Необходим, чтобы *web*-приложение было загружено и визуально отображалось для пользователя аналогично нативному приложению. Может содержать следующие элементы: *background_color*, *categories*, *description*, *dir*, *display*, *iarc_rating_id*, *icons*, *lang*, *name*, *scope*, *screenshots*, *serviceworker*, *short_name*, *start_url*, *theme_color* и пр. Все они отвечают за информацию, которую пользователь обычно видит после установки: название, цвет фона, создание иконки на экране смартфона и т. д.

Помимо этого, *PWA* требует, чтобы все ресурсы сайта передавались по *HTTPS*-протоколу.

На основе приведенной таблицы можно сделать вывод, что технология *PWA* самый быстрый способ создания приложения, если сайт уже есть и его необходимо дополнить функционалом использующим функции устройства пользователя, а также данными доступными оффлайн. *React Native или NativeScript* в свою очередь могут быть применены для создания *MVP* (*Minimum Viable Product*). *Apache Cordova* была одной из первых технологий для создания нативных приложений с использованием веб-технологий, поэтому закономерно, что он имеет большее количество весомых минусов перед современными технологиями и его применение на практике снижается.

Библиографические ссылки

1. Cordova [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Cordova (дата обращения 15.09.2021).

2. React [Электронный ресурс]. URL: https://ru.reactjs.org/ (дата обращения 15.09.2021).

3. React Native [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/React_Native (дата обращения 16.09.2021).

4. NativeScript [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/NativeScript (дата обращения 16.09.2021).

5. Прогрессивное веб-приложение [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/ Прогрессивное_веб-приложение (дата обращения 18.09.2021).

References

1. Cordova. Available at https://ru.wikipedia.org/wiki/Cordova (access 15.09.2021).

2. React. Available at https://ru.reactjs.org/ (access 15.09.2021).

3. React Native. Available at https://habr.com/ru/post/334380/ (access 16.09.2021).

4. NativeScript. Available at https://ru.wikipedia.org/wiki/NativeScript (access 16.09.2021).

5. Progressive Web Application. Available at https://ru.wikipedia.org/wiki/Прогрессивное_веб-приложение (access 18.09.2021).