УДК 621.3.049.77–048.24:537.2

ПЛАТФОРМА ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА JAVASCRIPT

Федосенко С.И.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Матюшков В.Е. – д-р техн.наук, профессор

Аннотация. React Native - это платформа JavaScript для написания мобильных приложений с рендерингом для iOS и Android. Он основан на React, JavaScript-библиотеке Facebook для создания пользовательских интерфейсов, но вместо ориентации на браузер он нацелен на мобильные платформы. Другими словами: веб-разработчики теперь могут писать мобильные приложения, которые выглядят, как нативные и все это с помощью библиотеки JavaScript. React Native упрощает одновременную разработку как для Android, так и для iOS. Принципы работы React Native практически идентичны React, за исключением того, что React Native не управляет DOM через виртуальную DOM. Он работает в фоновом режиме (который интерпретирует JavaScript, написанный разработчиками) непосредственно на конечном устройстве и взаимодействует с собственной платформой через сериализованные данные через асинхронный и пакетный мост.

Ключевые слова: react-native, web-development, кроссплатформенная разработка, мобильная разработка

Введение. Широкая распространенность, большое сообщество и гибкость языка программирования JavaScript позволяют многим компаниям экономить ресурсы на разработку программных решений. Кроссплатформенная разработка мобильных приложений позволяет существенно снизить затраты на команду, ведь нет необходимости в создании приложений под каждую операционную систему отдельно.

Согласно статистике StatCounter, доля пользователей с настольных устройств в феврале 2021 года составляет около 41%, в то время как мобильными устройствами пользуются более 54% во всем мире. Неудивительно, что популярность получения информации через мобильные устройства будет расти. Этому есть очень простое объяснение: люди изменяют свой образ жизни, проводят больше времени в интернете, чем когда-либо. Среднестатистический пользователь в среднем проводит в интернете около 7 часов в день и около половины этого времени – через мобильные устройства. Соответственно рост запросов на мобильную разработку будет расти с каждым годом, отсюда появляется спрос на высококвалифицированных специалистов и современные технологии, которые позволяют сэкономить бюджет на разработку мобильных приложений.

Нативные решения предполагают разработку под конкретную операционную систему (ОС). В настоящее время рынок поделен между Apple с операционной системой iOS и Google с Android. С одной стороны, данный способ открывает доступ ко всем нативным решениям той или иной ОС, предотвращает любые проблемы с производительностью, с другой стороны – необходимы большие финансовые и временные ресурсы на разработку одного и того же программного решения для разных ОС.

Кроссплатформенность – это способность программного обеспечения работать на двух и более платформах. Данная возможность позволяет разработчику писать код на одном языке программирования, при этом на выходе получится приложение под несколько платформ. Если речь идет о мобильной разработке, то это iOS и Android. На текущий момент многие крупные компании такие как Facebook, Instagram, Skype, Uber Eats, Walmart используют React Native.

В данной статье будут рассмотрены основные принципы и возможности построения мобильных приложений на языке JavaScript в том числе с помощью open-source решения React-Native. Очевидно, что для многих компаний данное программное решение позволит быстро и дешево выйти на рынок. В 2021 году популярность на данное решение будет расти.

Основная часть. Для реализации клиентской части программы используется JavaScript-библиотека React Native. Это мощный инструмент для реализации интерфейсов для пользователей. Цель данного инструмента – предоставить легкость и масштабируемость приложения, обеспечение вывода на экран всего того, что свойственно мобильным приложениям с высокой скоростью. React Native позволяет разработчикам моделировать состояние интерфейсов и декларативно описывать их. React Native – это всего лишь JavaScript, у библиотеки не очень большое API для изучения, всего несколько функций и способы их использования, что позволяет быстро изучить ее.

React Native выполняет рендеринг с использованием стандартных API рендеринга своей хост-платформы, отличает его от большинства существующих методов разработки кроссплатформенных приложений, таких как Cordova или Ionic. Существующие методы написания мобильных приложений используют комбинации JavaScript, HTML и CSS и обычно отображаются с использованием веб-представлений. Хотя этот подход может работать, он также имеет недостатки, особенно в отношении производительности. Кроме того, эти методы обычно не имеют доступа к набору собственных элементов пользовательского интерфейса хост-платформы. Когда эти фреймворки пытаются имитировать собственные элементы пользовательского интерфейса, результаты обычно кажутся немного неправильными [1].

React Native фактически переводит разметку в настоящие, нативные элементы пользовательского интерфейса, используя существующие средства рендеринга представлений на любой платформе, с которой работает разработчик. Кроме того, React работает отдельно от основного потока пользовательского интерфейса, поэтому приложение может поддерживать высокую производительность без ущерба для возможностей. Цикл обновления в React Native такой же, как и в React: при изменении свойств или состояния React Native повторно отображает представления. Основное различие между React Native и React в браузере заключается в том, что React Native делает это, используя библиотеки пользовательского интерфейса своей хост-платформы, а не разметку HTML и CSS. Для разработчиков, привыкших работать с React, это означает, что есть возможность писать мобильные приложения с производительностью и внешним видом нативного приложения, используя знакомые инструменты. React Native также представляет собой улучшение по сравнению с обычной мобильной разработкой в двух других областях: опыт разработчиков и потенциал кроссплатформенной разработки.

У всех компонентов React есть свои фазы. Когда экземпляр компонента создается и вставляется в DOM, он получает свойства, и с этого момента к ним можно получить доступ. Затем происходит жизненный цикл компонента [2]:

- монтирование (mounting);
- обновление (updation);
- размонтирование (unmounting);

Общая схема жизненного цикла компонента React представлена на рисунке 1.

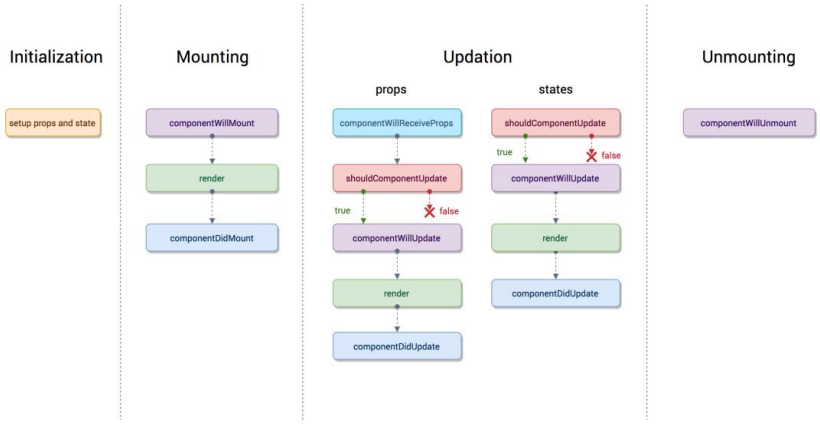

Рисунок 1 – Жизненный цикл компонента React

В разработке приложений на React Native применяется паттерн MVC. Исторически сложилось так, что шаблон MVC разделяет код на три отдельные части: модель, представление и контроллер. Основная цель этого шаблона - изолировать представление информации от взаимодействия с пользователем [3]. Основные понятия MVC:

– модель (model) – элемент управляет поведением и данными;
– отображение (view) – уровень представления модели в пользовательском интерфейсе;
– контроллер (controller) – производит манипуляции с моделью.

Простая модель MVC представлена на рисунке 2.

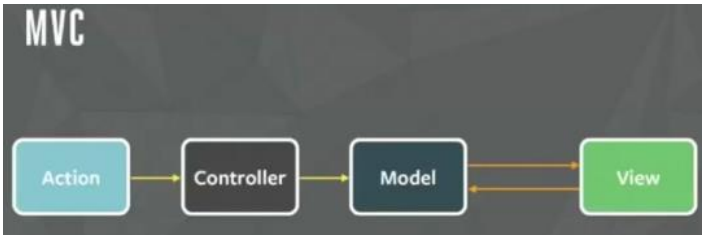

Рисунок 2 – Простая модель MVC

Компоненты представляют из себя: представление + обработка событий + состояние интерфейса.

Диаграмма на рисунке 3 показывает как фактически разделить стандартную MVC модель, чтобы получить компоненты. Выше линии осталось именно то, что пытается решить Flux: управление состоянием приложения и бизнес-логикой.

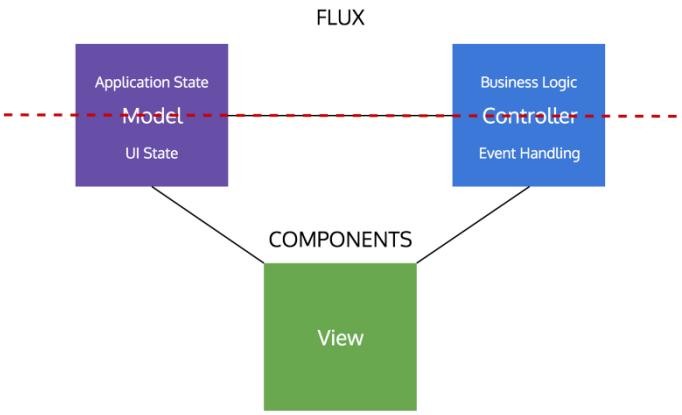

Рисунок 3 – MVC модель с использованием FLUX

Одновременно с популярностью React и компонентно-ориентированной архитектуры мы также увидели растущую популярность однонаправленной архитектуры для управления состоянием приложения.

Одной из причин их успешного совместного применения стало то, что они вдвоем полностью покрывают классический MVC подход. Они к тому же обеспечивают намного лучшее разделение ответственности при построении фронтенд-архитектуры.

Но это больше не история про один React Native. В Angular 2 можно увидеть точно такие же подходы, хотя и с различными вариантами управления состоянием.

MVC не мог сделать ничего лучше на клиенте. Он с самого начала был обречен на провал. Нам лишь нужно было время, чтобы это увидеть. Через этот пятилетний процесс фронтенд-архитектура эволюционировала в то, что мы видим сегодня. И если подумать, то пять лет не так уж много для того, чтобы выработать лучшие практики.

MVC был необходим в самом начале, потому что наши фронтенд-приложения становились все больше и сложнее, а мы не знали как их структурировать. MVC-подход справился со своим предназначением и заодно преподал хороший урок о том, как брать хорошую практику из одного контекста (сервера) и применять ее в другом (клиенте).

Таким образом, React отлично подойдет для проектов, где важна гибкость при создании больших экосистем, имеющие тенденцию разрастаться. Отлично подходит для командной разработки. UI-код легок и прост в сопровождении. Способен полностью взять на себя ответственность за уровень представления в архитектуре MVC (модель, представление, контроллер). Разработка пользовательского интерфейса по этому принципу – это современный подход в сфере предоставления данных. React Native позволяет использовать логику уже существующего веб-приложения, например корпоративного сайта, при создании мобильного приложения. Это означает, что разработчики могут использовать этот же самый код, который был использован в процессе создания сайта вместо того, чтобы начинать с чистого листа.

Заключение. Выполнено краткое описание популярного инструмента для разработки мобильных приложений на языке JavaScript.

Любое решение в сфере программирования имеет как свои плюсы, так и недостатки. В данном случае, React-Native имеет больше плюсов, чем минусов, которые, пожалуй, заключаются только лишь в небольших проблемах с быстродействием, однако, имеет ряд плюсов, из-за которых его всё чаще и чаще применяют в коммерческих разработках по всему миру.

Помимо более быстрой разработки, переиспользование кода позволяет избежать большого количества ошибок. Если вы создаете правильно спроектированные и удобные компоненты, которые планируете использовать в будущем снова, вам нужно будет писать меньше кода, когда вы решите создать с их помощью новый пользовательский интерфейс. Чем меньше нового кода вам нужно, тем меньше вероятность возникновения новых ошибок. К тому же, вы знаете ваши компоненты. Вы уже использовали и тестировали их при работе над реальным проектом, а значит при возникновении ошибок сможете предсказать причину их появления [4].

Благодаря переиспользованию кода стало гораздо проще создавать мобильные приложения. Код, который был написан во время работы на веб-приложением, может быть повторно использован для разработки мобильного приложения. Если вы планируете использовать не только сайт, но и мобильное приложение, нет необходимости нанимать две большие команды разработчиков.

Список литературы

1. *Learning React-Native / Bonnie Eisenman. – Beijing, Boston, Farnha, Sebastopol, Tokyo : OReilly, 2018. – 2 p.*
2. *Practical React Native / Frank Zammetti – USA // Apress. – 2018. – Vol. 1, N 14. – Pp. 25-26.*
3. *React Native for Mobile Development / Akshat Paul, Abhishek Nalwaya // Apress. – 2019. – Pp. 31–41.*
4. *Fullstack React Native The complete guide to React Native / Devin Abbott, Houssein Djirdeh, Anthony Accomazzo, Sophia Shoemaker // . – San Francisco : Fullstack.io. – 2017. – Vol. 1, N 24. – Pp. 47–62.*

UDC 621.3.049.77–048.24:537.2

PLATFORM FOR DEVELOPING MOBILE APPLICATIONS ON JAVASCRIPT

Fedosenko S.I.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus (style T-institution)

Matyushkov V.E. - Doctor of Technical Sciences, Professor

Annotation. React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It's based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly «native», all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS. The working principles of React Native are virtually identical to React except that React Native does not manipulate the DOM via the Virtual DOM. It runs in a background process (which interprets the JavaScript written by the developers) directly on the end-device and communicates with the native platform via serialized data over an asynchronous and batched bridge.

Keywords. react-native, web-development, cross platform development, mobile development