Мобильные среды веб-разработки для мобильных приложений

Сорока Антон Сергеевич – студент, кафедра компьютерной инженерии, Донецкий национальный технический университет, г. Донецк

Завадская Татьяна Владимировна- доцент, кафедра компьютерной инженерии, Донецкий национальный технический университет, г. Донецк

Аннотация

Сорока А.С., Завадская Т.В. Мобильные среды веб-разработки для мобильных приложений

Статья посвящена анализу современных мобильных сред web-приложений. Проведен сравнительный анализ средств наиболее подходящих для реализации конкретных задач.

Ключевые слова: React Native, Android, кроссплатформенность, Cordova, нативные приложения.

Введение. Каждый пользователь имеет свое собственное мобильное устройство и организации, занимающиеся разработкой программного обеспечения для мобильных устройств, понимают, что большую часть свободного времени люди используют их. Исходя из этого создаются различные приложения, что приводит к росту потребности мобильных сред разработки. В связи с этим можно наблюдать появления множества платформ для web-разработки, из-за чего разработчику трудно остановить свой выбор на конкретной платформе.

Постановка проблемы. Существует множество сред для разработки, но самые востребованные из них – это те, которые компилируются в JavaScript (JS). Большую популярность они приобрели из-за низкого порога вхождения, наличия кроссплатформенности, возможности использования web-элементов, нативности. Одни из популярных сред (рис.1 основные среды разработки): React Native, Apache Cordova (ранее PhoneGap).

Рисунок 1 - Среды разработки

Нативные приложения, кроссплатформенность. Рассматриваемые среды согласно документации, дают возможность разработки нативных и кроссплатформенных приложений. Прежде чем рассматривать их нужно дать им определения.

Нативные приложения — это прикладные программы, которые были разработаны для использования на определённой платформе или на определённом устройстве. Одно из преимуществ нативных приложений — то, что они оптимизированы под конкретные операционные системы, поэтому они могут работать корректно и быстро [3].

Кроссплатформенность — способность программного обеспечения работать с двумя и более аппаратными платформами и (или) операционными системами (ОС). Это обеспечивается благодаря использованию высокоуровневых языков программирования, сред разработки и выполнения, поддерживающих условную компиляцию, компоновку и выполнение кода для различных платформ [4].

Из этого следует, что разработанное приложение может верно функционировать на нескольких ОС, которые можно будет разместить в Play маркете или App Store. Ключевой составляющей таких приложений является то, что ресурсы приложения используют нативный компонент и выполняются в WebView. Данный способ позволяет устанавливать связь со всеми доступными элементами устройства: камера, акселерометр,

динамики, кнопки, микрофон и т.д. Возможности таких приложений ограничены только скоростью обработки рендера. Остановимся конкретней на выбранных средах разработки.

React Native. React Native разработана для создания нативно отображаемых iOS и Android-приложений. В его основе лежит разработанная в Facebook JS-библиотека React, предназначенная для создания пользовательских интерфейсов. Данный фреймворк ориентирован не на браузер, а на мобильные платформы. При разработке приложения можно использовать React Native для написания чистых, быстрых мобильных приложений, используя привычную JS-библиотеку React и единой кодовой базой JavaScript. Прежде чем рассматривать детально React Native, определим, что такое React.

React — это JS-библиотека для создания пользовательских интерфейсов, для веб-приложений. React широко распространен, и в отличие от более крупных MVC-фреймворков решает относительно узкую задачу: рендеринг интерфейса. React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как Redux [1,2].

В React компонент описывает собственное отображение, а затем библиотека обрабатывает рендеринг. В случае, если нужно отрисовать компоненты для web, то React использует стандартные HTML-тэги. Благодаря тому же уровню абстракции — для рендеринга в iOS и Android React Native вызывает соответствующие *Application Programming Interface* (API). В iOS компоненты отрисовываются в UI-виды, а в Android — в нативные виды. У React Native есть ряд отличий:

- стандартный div заменяется View;
- тег img заменяет Image
- доступны спец-теги для различных платформ

React Native запускает приложение с помощью JS-движка хост-платформы, без блокирования основного UI-потока. В итоге разработчик получает преимущества нативной производительности, анимации и поведения без необходимости писать на Objective-C или Java. Другие методы разработки кроссплатформенных приложений, вроде Cordova или Titanium, никогда не достигнут такого уровня нативной производительности или отображения [1]. Для написания кода используется JSX (пример на рис.2), который похож на стандартный HTML, CSS, JavaScript.

Рисунок 2 – пример JSX кода

В отличие от стандартной разработки под iOS и Android, React Native имеет больше преимуществ, т.к. приложение в основном состоит из JavaScript. Разработчик может пользоваться многочисленными достоинствами web-разработки. Например, чтобы увидеть внесённые в код изменения, можно мгновенно «обновить» приложение вместо длительного ожидания завершения традиционного ребилда. Большинство API во фреймворке — кроссплатформенные, так что достаточно написать компонент React Native, и он будет работать в iOS и Android. React Native позволяет назначать платформозависимые версии каждого компонента, которые можно потом интегрировать в своё приложение [1].

У React Native есть свои недостатки:

- относительно новый проект, которому свойственны проблемы новых библиотек;
- отсутствуют некоторые функции;
- не выработаны оптимальные методики использования;
- от релиза к релизу внедряются серьёзные изменения, которые не всегда влияют на работу положительно.

Из преимуществ можно выделить следующие:

- возможность использования единой кодовой базы под iOS и Android;
- низкий порог вхождения;
- быстрый цикл разработки;
- возможность получения API с собственного сайта;
- отсутствие свойственных фреймворку ограничений.

Подводя итог, React Native обеспечивает высококачественную кроссплатформенную мобильную разработку, но не подходит для разработки крупных игровых приложений.

Apache Cordova. Apache Cordova — это платформа разработки мобильных приложений с открытым исходным кодом. Она позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кросс платформенной разработки, избегая встроенного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри компонента нацеленною на каждую платформу и используют стандартные API для доступа к датчикам устройства, данным и состоянию сети [5].

Приложение реализовано как веб-страница, где по умолчанию главным файлом является index.html, который ссылается на CSS, JS, изображения, файлы мультимедиа или другие ресурсы, необходиме для его запуска. Выполняется это все как WebView в пределах оболочки приложения, которая будет доступны в магазины приложений.

WebView с поддержкой Cordova может представлять приложения и его пользовательский интерфейс. Поддерживаются следующие платформы:

- Android;
- iOS;
- WP8;
- Windows;
- Ubuntu;
- Blackberry 10.

В работе с некоторыми платформами Cordova может выступвть в качестве компонента в больших, гибридных приложениях, которые объединяются в WebView с другими компонентами приложения.

Интерфейс плагина доступен для Cordova и других компонентов, для повторного использования кода. Это позволяет вызывать код на языке платформы из JavaScript. На нескольких платформах устройств согласуются JS API, чтобы этот машинный код скомпилировать. Сторонние плагины предоставляют дополнительные привязки для функции, которые могут не выполняться на всех платформах. Можно найти эти аналоги таких плагинов в реестре и использовать их в своем приложении. На рис. 3 представлен пример реализации разметки и кода в данной среде.

Рисунок 3 – Пример кода и разметки

Cordova не предоставляет каких-либо виджетов для пользовательского интерфейса или MV-фреймворков. Если нужно использовать UI-виджеты и/или MV-фреймворк, то необходимо выбрать их и включать в приложение самостоятельно, как ресурсы третьей стороны [5].

Как и у любой среды у Cordova есть свои достоинства и недостатки. Из достоинств можно выделить следующие:

- простата изучения;
- доступ к нативному функционалу (камера, динамики, кнопки и т.д.);
- простое API;
- возможность использования JavaScript библиотек.

Также имеется следующие недостатки:

- плохая документация;
- низкая происводительность по сравнению с другими нативными приложениями;
- отсутствие готовых элементов пользовательского интерфейса;
- большое количество использования плагинов.

Данный фреймворк может подойти для разработки небыстрых приложений, которые работают с небольшим объемом данных, но при этом нужна большая кроссплатформенность и быстрота создания приложения.

Заключение. Технологии web-разработок развиваются очень быстро и не стоит останавливается на одном способе разработки. Во многих случаях есть возможность комбинирования нескольких способов создания приложения. Например, Cordova может использовать библиотеку React, но такой способ может влиять на производительность. Пока в мире IT индустрии нет определенных стандартов, и предписаний указывающих какую технологию использовать, поэтому следует выбрать технологию, которая индивидуально подойдет к проекту. Выбранная технология должно позволять выполнить, поставленную перед разработчиком задачу с максимальной эффективностью.

ЛИТЕРАТУРА

1. Создание кроссплатформенных приложений с помощью React Native [Электронный ресурс]: habr— Режим доступа: https://habr.com/ru/company/nixsolutions/blog/324562/

2 .React Native [Электронный ресурс]: facebook.github — Режим доступа: https://facebook.github.io/react-native/docs/tutorial.html

3. Нативные приложения [Электронный ресурс]: wikipedia — Режим доступа:

https://ru.wikipedia.org/wiki /Нативные приложения

4. Кроссплатформенность [Электронный ресурс]: wikipedia — Режим доступа:

https://ru.wikipedia.org/wiki/ Кроссплатформенность

5. Введение Apache Cordova [Электронный ресурс]: Cordova — Режим доступа:

https://cordova.apache.org/docs/ru/latest/guide/overview/