

Рисунок 8 – дерево рекурсивных вызовов для тестовых данных

Время работы

Ожидаемая сложность алгоритма в лучшем случае составляет $O(n \cdot \log(n))$, это когда номер наиболее удаленной точки всегда оказывается лексикографически центральным. Однако, в худшем случае сложность алгоритма составляет $O(n^2)$. Это достигается, например, в случае, если номер наиболее удаленной точки всегда соседний к номеру граничной точки [1].

Литература

1. Будяков В.В., Рыбалко К.К., Долженко А.М. Сравнительный анализ технологий построения виртуальных миров // Современные тенденции развития и перспективы внедрения инновационных технологий в машиностроении, образовании и экономике. 2016. Т. 2. № 1. С. 285-291.
2. Долженко А.М., Костыркин А.В., Анохин Д.В. Применение карт Google в прикладных программах // Современные тенденции развития и перспективы внедрения инновационных технологий в машиностроении, образовании и экономике. 2016. № 1. С. 181-183.
3. Келлер А.В., Захарова Е.В. Задача оптимального измерения с учетом резонансов: алгоритм программы и вычислительный эксперимент // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2012. № 27. С. 58-65.
4. Новик Н.В. Информационные технологии как средство повышения эффективности профессиональной подготовки инженера (на материалах дисциплины «Инженерная графика») // Общество: социология, психология, педагогика. 2016. № 8. С. 88-90.
5. Фоменко А.Е., Чурсин А.И. Управление территориями при помощи геоинформационных технологий // Международный журнал прикладных и фундаментальных исследований. 2015. № 5-1. С. 80-82.

УДК 004.42 : 004.896

МОБИЛЬНЫЕ СРЕДЫ WEB-РАЗРАБОТКИ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Сорока Антон Сергеевич, Завадская Татьяна Владимировна

Донецкий национальный технический университет,

г. Донецк, Донецкая народная республика

Аннотация

Статья посвящена рассмотрению современных мобильных сред web-приложений. Проведен их сравнительный анализ с точки зрения применения для реализации конкретных задач. Выбранная технология или набор технологий должны обеспечить выполнение поставленной перед разработчиком задачи с максимальной эффективностью.

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

MOBILE WEB DEVELOPMENT ENVIRONMENTS FOR MOBILE APPS

Soroka Anton Sergeevich, Zavadskaya Tatyana Vladimirovna

Donetsk National Technical University, Donetsk,
Donetsk People's Republic

Abstract

The article is devoted to the consideration of modern mobile web-application environments. Their comparative analysis from the point of view of application for realization of concrete tasks is carried out. The chosen technology or set of technologies should ensure the fulfillment of the task set for the developer with maximum efficiency.

Keywords: *React Native, Android, cross-platform, Cordova, native applications.*

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 появление 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. Примерами могут служить следующие среды разработки: React Native, Apache Cordova (ранее PhoneGap). Проблемой является выбор одной или нескольких сред разработки, обеспечивающих наиболее эффективное решение поставленной перед разработчиками задачи.

Нативные приложения, кроссплатформенность

Рассматриваемые среды, согласно документации, дают возможность разработки нативных и кроссплатформенных приложений. 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. 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.

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 натив-

ный 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.

В React компонент описывает собственное отображение, а затем библиотека обрабатывает рендеринг. В случае, если нужно отрисовать компоненты для web, то React использует стандартные HTML-теги. 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-потoka. 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.

Для написания кода используется JSX (пример на рис.1), который похож на стандартный HTML, CSS, JavaScript. 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.

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; воз-

возможность получения API с собственного сайта; отсутствие свойственных фреймворку ограничений.

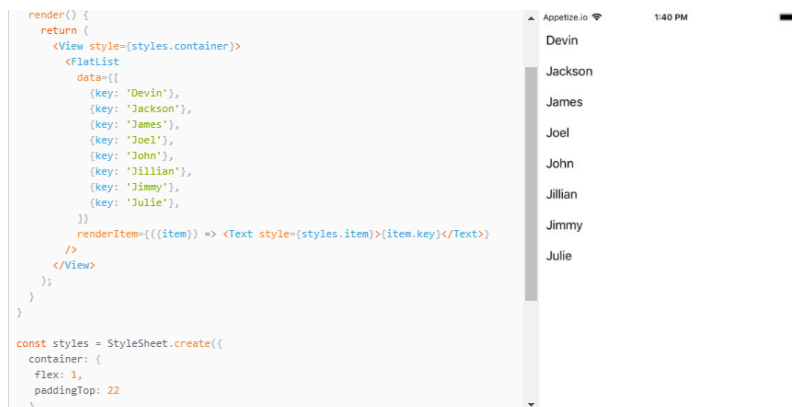


Рисунок 1 – Пример JSX кода

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 нацеленного на каждую платформу и используют стандартные API для доступа к датчикам устройства, данным и состоянию сети.

Приложение реализовано как веб-страница, где по умолчанию главным файлом является index.html, который ссылается на CSS, JS, изображения, файлы мультимедиа или другие ресурсы, необходимые для его запуска. Выполняется это все как WebView в пределах оболочки приложения, которая будет доступна 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 выступать 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. 2 представлен пример реализации разметки и кода в данной среде.

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. Как и у любой среды у Cordova есть свои достоинства и недостатки. Из достоинств можно выделить следующие: простота изучения; доступ к нативному функционалу (камера, динамики, кнопки и т.д.); простое API; возможность использования JavaScript библиотек.

```
<!DOCTYPE html>
<html>
<head>
<title>Compass Example</title>
<script type="text/javascript" charset="utf-8" src="cordova.js"></script>
<script type="text/javascript" charset="utf-8">
var watchID = null;
function onDeviceReady() {
startWatch();
}
function startWatch() {
var options = { frequency: 3000 };
watchID = navigator.compass.watchHeading(onSuccess, onError, options);
}
</script>
</head>
<body>
<div id="heading">Waiting for heading...</div>
<button onclick="startWatch();">Start Watching</button>
<button onclick="stopWatch();">Stop Watching</button>
</body>
</html>
```

Рисунок 2 – Пример кода и разметки

Также имеются следующие недостатки: плохая документация; низкая производительность 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.

Выводы

Технологии web-разработок развиваются очень быстро, поэтому не стоит останавливаться 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, следует выбирать технологию, которая индивидуально подойдет к проекту. Выбранная технология должна позволять выполнить поставленную перед разработчиком задачу с максимальной эффективностью.

УДК 004.738.2

تأثير الارتباط المكاني وعدد اختيار الهوائيات على سعة قناة MIMO

(Influence of spatial correlation and antenna selection number on the channel of capacity MIMO)

Al Taher I.A.

Department of Electronics, radio engineering and communication systems, Don State Technical University, Shakhty, Russian Federation

الخلاصة (Abstract)

في هذا البحث نقترح طريقة اختيار الهوائي في قناة متعددة الادخال- متعددة الاخراج (MIMO) تعمل على تحسين سعة قناة هوائيات الإرسال المتعددة ، التي يتم الحصول عليها من خلال اختيار أفضل هوائي في الإرسال. وكذلك يمكن استخدام هذه الطريقة لتقليل الطاقة المستهلكة في التردد الراديوي (Radio Frequency) RF في الهوائيات ، للحصول على كفاءة طاقة عالية لتحقيق أقصى سعة قناة للمنظومة (MIMO) ، وكذلك دراسة تأثير قناة الارتباط المكانية spatial correlation () على سعة الإرسال المتعددة. في هذا البحث ، أظهرت النتائج أن نسبة الإشارة إلى الضوضاء (Signal to Noise Ratio) SNR واحتمالية الخطأ في عدد البتات (Bit Error Rate) BER عند SNR تساوي (18) ديسيبل (dB) هي 10^{-3} ، وكذلك زيادة معامل الارتباط المكانية تؤدي الى تقليل سعة القناة بسبب الخسائر في عناصر الهوائي بين الإرسال والاستقبال ، عندما تكون نسبة الإشارة إلى الضوضاء هي (18) ديسيبل (dB) بين هوائيات الاستقبال.