УДК 004.43

СРАВНЕНИЕ СОВРЕМЕННЫХ СРЕДСТВ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Чуриков Е.А., Зудилова Т.В., Ананченко И.В., Осипов Н.А., Иванов С.Е., Осетрова И.С.

ФГАОУ ВО «Национальный исследовательский университет ИТМО», Санкт-Петербург, e-mail: churikov.egor@bk.ru, zudilova@ifmo.spb.ru, anantchenko@yandex.ru, nikita@ifmo.spb.ru, serg_ie@mail.ru, irina@ifmo.spb.ru

Рассматриваются ключевые различия между современными средствами разработки мобильных приложений для определения параметров, существенно влияющих на скорость, удобство и стоимость процесса разработки мобильных приложений. Цель – проанализировать характеристики существующих и востребованных на рынке средств разработки мобильных приложений, влияющие на разработку, дать рекомендации о применимости технологий. Рассматриваются как нативные средства разработки мобильных приложений (Android SDK, iOS SDK), так и кросс-платформенные фреймворки (Flutter, React Native, Ionic и другие), характеристики средств разработки как способ отрисовки компонентов, возможности взаимодействия с операционной системой, итоговый размер приложения и производительность. Сравнение включает в себя не только сравнение технических характеристик, но и обобщенный коммерческий опыт использования данных технологий, поскольку ни одна технология не существует в вакууме и должна быть применима к реальным проектам. Выполненное сравнение фреймворков видится полезным для разработчиков, так как выбор средства разработки является, пожалуй, важнейшим моментом в реализации мобильного приложения. В случае неоптимального выбора SDK часть планируемого функционала может быть нереализуема по техническим причинам. Кроме того, неверный выбор SDK может повлиять на дальнейшее развитие приложения – сложности в поиске исполнителей или в отставании SDK от последней версии операционной системы.

Ключевые слова: мобильные приложения, разработка, Mobile, Android, iOS, SDK, Flutter, React Native, NativeScript

COMPARISON OF MODERN MOBILE APPLICATION DEVELOPMENT TOOLS

Churikov E.A., Zudilova T.V., Ananchenko I.V, Osipov N.A., Ivanov S.E., Osetrova I.S.

ITMO National Research University, Saint-Petersberg, e-mail: churikov.egor@bk.ru, zudilova@ifmo.spb.ru, anantchenko@yandex.ru, nikita@ifmo.spb.ru, serg_ie@mail.ru, irina@ifmo.spb.ru

The key differences between modern and market-demanded mobile application development tools are considered to determine the parameters that significantly affect the speed, convenience and cost of the mobile application development process. The purpose of the performed research is to analyze the characteristics of existing and demanded mobile application development tools on the market that affect the development, to make recommendations on the applicability of technologies. Both native mobile application development tools (Android SDK, iOS SDK) and cross-platform frameworks (Flutter, React Native, Ionic and others), the characteristics of development tools as a way of rendering components, the possibilities of interaction with the operating system, the final application size and performance are considered. The comparison includes not only a comparison of technical characteristics, but also a generalized commercial experience of using these technologies, since no technology exists in a vacuum and should be applicable to real projects. The comparison of frameworks is seen as useful for developers, since the choice of a development tool is perhaps the most important moment in the implementation of a mobile application. In the case of a non-optimal SDK choice, part of the planned functionality may not be feasible for technical reasons. In addition, the wrong choice of SDK may affect the further development of the application – difficulties in finding performers or the SDK lagging behind the latest version of the operating system.

Keywords: mobile, android, iOS, SDK, Flutter, React Native, Xamarin, Ionic, Cordova, PhoneGap, NativeScript

В настоящее время на рынке разработки мобильных приложений нет жесткого стандарта инструмента (языка или фреймворка) для создания приложений. Некоторые компании предпочитают сэкономить время и деньги на разработку мобильного приложения и выбирают одно из кросс-платформенных решений, которых в данный момент на рынке существует больше десятка, другие (чаще всего более состоятельные) компании предпочитают нативную разработку, чтобы получить полный доступ ко всем функциям операционных систем. Были рассмотрены наиболее популярные [1] инструменты создания клиентских мобильных приложений, такие как кросс-платформенная разработка с использованием фреймворков React Native [2], Xamarin [3], NativeScript [4], Flutter [5], Ionic, Cordova [6], PhoneGap [7] и нативная разработка под операционные системы iOS [8] и Android [9].

Целью выполненного исследования является классификация популярных среди разработчиков средств разработки мобильных приложений, сравнение нативных и ненативных (кросс-платформенных) решений как по техническим характеристикам (таким

как используемый язык программирования, подход к созданию компонентов, целевые операционные системы и др.), так и по бизнес-требованиям (например, легкость поиска разработчика под технологию, год выпуска инструмента, открытость исходного кода), а также выдача рекомендаций по выбору инструмента разработки под проектные требования, бюджет и имеющееся время на разработку мобильного приложения.

Обзор современных средств мобильной разработки. *Нативная iOS-разработка* подразумевает под собой разработку мобильных приложений под операционную систему iOS, созданную компанией Apple, с использованием языка программирования Swift, спроектированного в той же компании. Разработка и отладка возможна исключительно на компьютерах с операционной системой MacOS, легальное использование которой подразумевает покупку MacBook. Чтобы приложение можно было загрузить в магазин приложений AppStore (единственный легальный способ установки мобильных приложений в операционной системе iOS – загрузка из официального магазина приложений Apple AppStore), разработчику нужно иметь Apple Developer Account, покупка которого на год обойдется в 99$. Как можно заметить, финансовый порог старта в iOS-разработку довольно высок для начинающих разработчиков. Однако крупные компании могут себе позволить взять все эти траты на себя и, наняв хороших специалистов, сделать приложение с глубокими платформенными интеграциями для получения доступа к самым последним функциям операционной системы iOS и наилучшей производительности на этой платформе [8].

Нативная Android-разработка подразумевает под собой разработку мобильных приложений под операционную систему Android, принадлежащую компании Google, с использованием языка программирования Kotlin. Такой тип разработки позволяет реализовать в мобильном приложении самые сложные и глубокие интеграции с операционной системой Android для добавления сложного функционала и получения наилучшей производительности на этой платформе. Стоимость регистрации аккаунта разработчика составляет 25$ (оплачивается единоразово и не требует платного пролонгирования), а вести разработку можно на компьютерах с ОС Windows, MacOS, Linux и ChromeOS [9].

Фреймворки для кросс-платформенной разработки мобильных приложений были созданы как альтернатива нативной разработке. Главными их преимуществами перед нативными инструментами являются скорость и стоимость разработки. Среди недостатков может оказаться низкая производительность или недостаточно широкий функционал для интеграции с операционными системами. Рассмотрим более детально такие кросс-платформенные фреймворки, как React Native [2], Xamarin [3], NativeScript [4], Flutter [5], Ionic, Cordova [6], PhoneGap [7]. Unity не рассматривается ввиду области его применения (индустрия мобильных игр), остальные фреймворки не рассматриваются ввиду их непопулярности среди разработчиков. Стоит отметить, что при загрузке в магазины App Store или Google Play приложения, созданного с помощью одного из кросс-платформенных фреймворков, также требуется иметь действующий аккаунт разработчика на соответствующей платформе.

Информация о популярности вышеперечисленных фреймворков среди разработчиков взята с сайта statista.com. Согласно опросу разработчиков 2021 года, Flutter – самый популярный кросс-платформенный мобильный фреймворк, используемый разработчиками по всему миру. Согласно опросу, 42% разработчиков программного обеспечения использовали Flutter. В целом примерно треть мобильных разработчиков используют кросс-платформенные технологии или фреймворки; остальные мобильные разработчики используют нативные инструменты [1]. На рисунке 1 представлена диаграмма популярности кросс-платформенных фреймворков среди разработчиков.

Классификация кросс-платформенных решений. Несмотря на то что все кросс-платформенные фреймворки подразумевают под собой написание кода и дальнейшее его преобразование в установочные файлы для разных операционных систем, у отмеченных выше фреймворков довольно много различий. Классифицировать их можно, например на основе работы исполняемого кода или по способу описания интерфейса. На рисунке 2 представлена классификация кросс-платформенных фреймворков по подходу, применяемому при разработке.

Нативный User Interface (далее – UI), ***общий код*** (React Native, Xamarin, NativeScript [2-4]). Фреймворки такого типа подойдут в первую очередь тем, кто хочет добиться, чтобы мобильное приложение выглядело подобно нативному. Фреймворки, следующие данному подходу, можно также классифицировать разными способами.

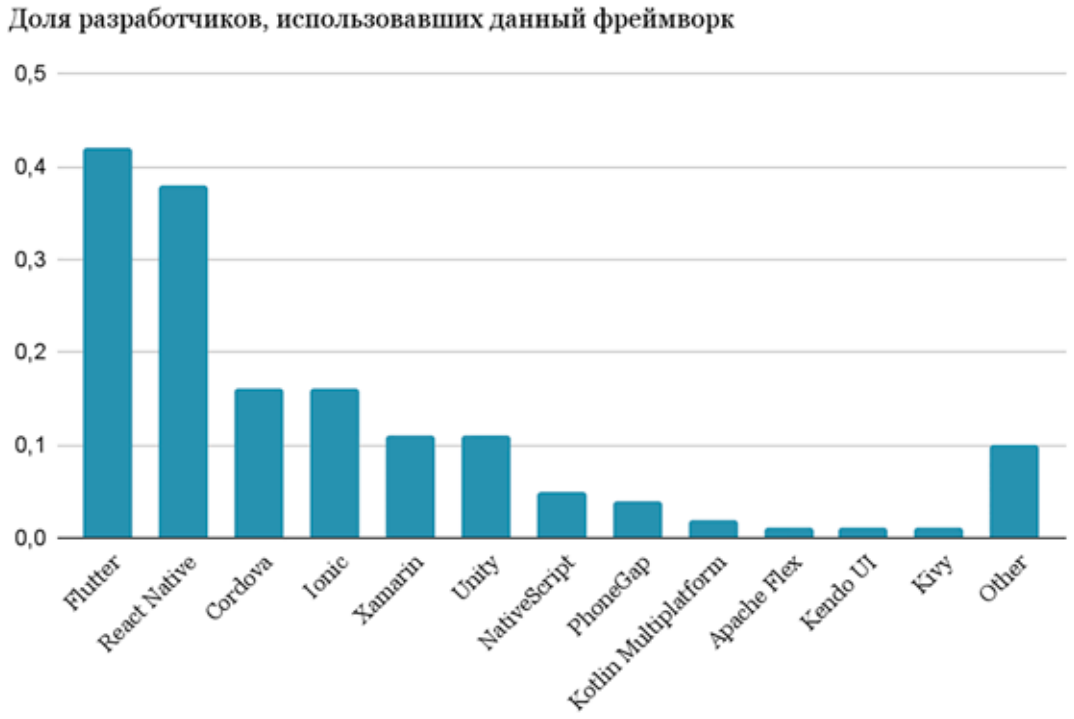

Рис. 1. Популярность фреймворков кросс-платформенной разработки мобильных приложений среди разработчиков

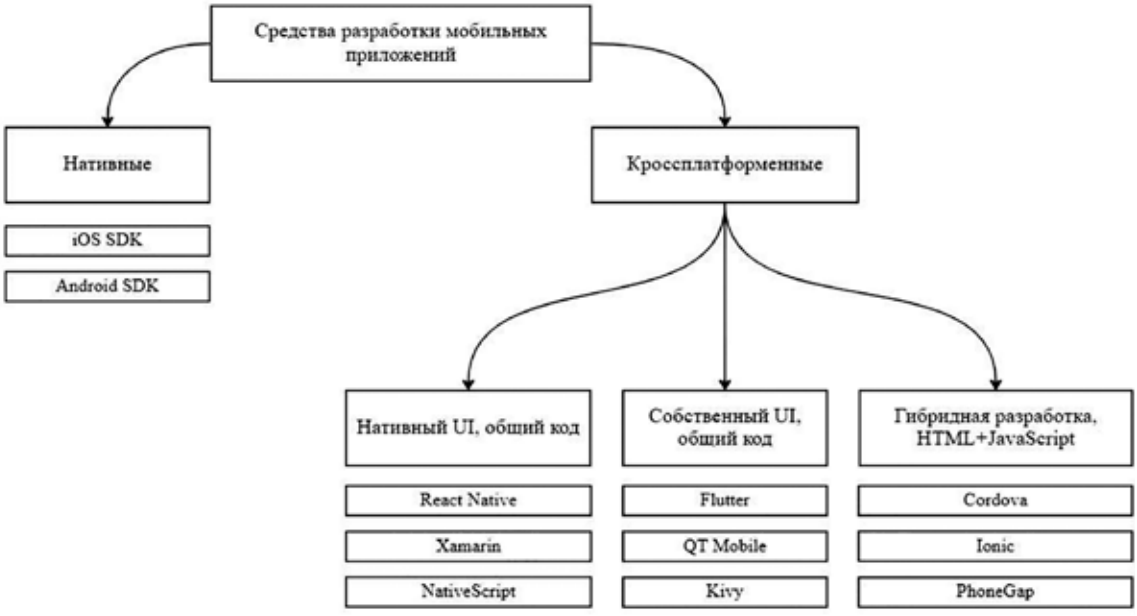

Рис. 2. Классификация средств разработки мобильных приложений

Классификация по работе исполняемого кода. Код может быть компилируемым (как в случае фреймворка Xamarin и используемого в нем языка C#), так и интерпретируемым с JIT-компиляцией (большинство подобных фреймворков использует JavaScript).

Классификация по способу описания интерфейса. Некоторые фреймворки (на-

пример, Xamarin) используют нативные элементы интерфейса, в то время как другие (например, Xamarin Forms) реализуют универсальные элементы интерфейса – набор виджетов, который преобразуется в подходящие компоненты интерфейса каждой платформы. Также существует способ, при котором реализуется разный интерфейс для разных платформ с сохранением общего подхода (например, React Native создает обёртки вокруг нативных элементов интерфейса. Соответственно, интерфейс описывается для каждой платформы отдельно, но по одному принципу).

Собственный UI, общий код (например, Flutter [5]) – фреймворки используют единую кодовую базу и отрисовывают компоненты пользовательского интерфейса собственными силами с помощью графических движков (например, движок рендеринга skia в фреймворке Flutter). Фреймворки такого типа подойдут в первую очередь тем, кто хочет добиться высокой производительности и реализации сложных дизайнерских интерфейсов. Из минусов данного подхода можно отметить заметно больший вес приложений по сравнению с нативными (а также более высокое потребление оперативной памяти устройства) и одинаковый внешний вид приложения на устройствах разных ОС (в каких-то случаях это может не быть проблемой).

Гибридная разработка HTML + JavaScript (Ionic, Cordova, PhoneGap [6; 7]). Данный подход подразумевает под собой, по сути, веб-страницу, открытую во встроенном в приложение браузере. Главный плюс такого подхода – минимальная стоимость разработки, поскольку можно не только повторно использовать код, написанный для веб-страниц, но и задействовать разработчиков, участвовавших в создании данного веб-приложения. Также важным плюсом данного подхода является большое количество библиотек, написанных для HTML + JS, что означает отсутствие проблем, связанных с необходимостью создавать собственную версию известной библиотеки. Однако даже этих весомых преимуществ будет недостаточно, чтобы перекрыть такие существенные недостатки, как отсутствие поддержки нативных жестов и проблема совместимости веб-браузеров. Если первая проблема очевидна, то про вторую стоит рассказать более детально. На старых версиях Android (до версии 5) WebView был частью платформы и не обновлялся автоматически. Поэтому гибридные приложения не могут использовать новейшие функции браузера на этих устройствах. Следовательно, гибридные приложения ограничивают минимальную версию Android (в настоящее время около 13% устройств удалены с рынка) или включают WebView в код приложения (проект CrossWalk), увеличивая размер приложения на десятки мегабайт. Поэтому использовать данный подход рекомендуется исключительно в условиях очень сильно ограниченного бюджета или при отсутствии необходимости делать полнофункциональное мобильное приложение, ограничиваясь повторением функционала существующего веб-сайта.

Лидеры рынка. Сравним двух лидеров рынка кросс-платформенной мобильной разработки – React Native и Flutter. Фреймворки, реализующие подход гибридной HTML+JS разработки, не рассматриваются ввиду вышеперечисленных причин, Xamarin не рассматривается ввиду более сильного конкурента в лице React Native.

React Native – фреймворк для разработки мобильных приложений под операционные системы iOS и Android, созданный компанией Facebook в 2015 [2]. Данный фреймворк реализует подход «Нативный UI, общий код», что позволяет разработчикам получить доступ к нативным компонентам обеих операционных систем, при этом логика мобильного приложения пишется на языке JavaScript. Для взаимодействия с платформой React Native использует так называемый мост. В то время как JavaScript и нативные потоки написаны на совершенно разных языках, эта функция моста делает возможной двунаправленную связь. Это означает, что если уже есть собственное приложение для iOS или Android, то все равно можно использовать его компоненты или перейти к разработке на React Native. Также React Native – технология с открытым исходным кодом, что позволяет разработчикам со всего мира участвовать в развитии платформы. Сообщество разработчиков также достаточно большое. React Native отлично подойдет, если стоит задача быстро и дешево сделать приложение для iOS и Android, которое по своему внешнему виду не будет отличаться от нативного (так как будет состоять из нативных компонентов). Однако из-за использования JavaSctipt существуют некоторые проблемы с производительностью, что можно заметить при добавлении в проект сложных анимаций.

Flutter – фреймворк, созданный компанией Google в 2017 [5]. С помощью Flutter можно создавать приложения для iOS, Android, MacOS, Windows, Linux, а также веб-приложения. Flutter-приложения пишутся с использованием языка программирования Dart, также изобретенного в Google и позиционируемого как более зрелая

альтернатива JavaScript. Главной особенностью данного фреймворка является то, что все компоненты пользовательского интерфейса наследуются от одного класса Widget, будь то кнопка, текст или всё приложение, также можно создавать собственные виджеты, что делает возможным построение интерфейсов любой сложности. Flutter компилируется в нативный код под каждую из платформ, что гарантирует достаточно высокий уровень производительности. «Под капотом» он использует Skia в качестве графического движка. Все библиотеки, доступные в нативных приложениях SDK, и платформенные API могут быть использованы для Flutter-приложений с помощью технологии MethodChannel. Исходный код Flutter также открыт, а размер сообщества разработчиков уже превысил React Native. Flutter станет отличным выбором для приложения с необычным пользовательским интерфейсом и сложными анимациями, однако «из коробки» не поддерживаются нативные UI-компоненты, что станет очевидным минусом, если приложение должно выглядеть полностью нативно.

Таблица 1

Характеристики нативных инструментов разработки мобильных приложений

	iOS	Android
Язык	Swift	Kotlin
Приложения	Нативные	Нативные
Релиз	2014	2011
Разработчик	Apple	Google
Платформы	iOS	Android
Инструменты создания UI	Нативные компоненты	Нативные компоненты
Производительность	Максимально высокая на данной платформе	Максимально высокая на данной платформе
Open Source	да	да

Таблица 2

Характеристики популярных средств разработки кросс-платформенных мобильных приложений

	Flutter	React Native	Xamarin	Ionic	PhoneGap	NativeScript	Cordova
Язык	Dart	JS	C#	HTML, CSS, JS	HTML, CSS, JS	JS	HTML, CSS, JS
Приложения	Свой UI, свой код	Нативный UI, общий код	Нативный UI, общий код	Гибрид-веб	Гибрид-веб	Нативный UI, общий код	Гибрид-веб
Разработчик	Google	Facebook	Microsoft	Drifty Co.	Nitobi Software	Telerik	Adobe Systems
Open Source	да	да	да + платные пакеты	да + платные пакеты	да	да	да
Инструменты создания UI	Собственные виджеты	Нативные компоненты + декларативный UI	Xamarin. IOS/Android + Xamarin. Forms	HTML/CSS	HTML/ CSS	Нативные компоненты	HTML/ CSS
Производительность	Очень высокая	Высокая	Высокая	Средняя из-за веб-технологий	Средняя из-за веб-технологий	Высокая	Средняя из-за веб-технологий
Релиз	2017	2015	2011	2013	2005	2014	2009
Платформы	IOS, Android, Web, Desktop	IOS, Android, UWP	IOS, Android, UWP	IOS, Android, Web	IOS, Android	IOS, Android	IOS, Android

Подведение итогов. Выбор инструмента разработки – важнейший этап в разработке мобильного приложения. Крупные компании могут позволить себе разрабатывать одно и то же приложение под каждую платформу с использованием нативных инструментов, получая при этом полный набор функций операционной системы, максимальную производительность и нативные компоненты (в таблице 1 приведены важные характеристики нативных инструментов разработки мобильных приложений). Другие компании, пытаясь удешевить разработку, прибегают к использованию кросс-платформенных решений, выигрывая в стоимости и сроках разработки, но получая более низкую производительность (хотя многие разработчики фреймворков и заявляют о производительности, близкой к нативной, при использовании их решений), ненативный внешний вид (как в случае с Flutter и Web-гибридными приложениями) или отставание по инструментарию от последних версий операционной системы. В таблице 2 сравниваются важные характеристики популярных фреймворков для кросс-платформенной мобильной разработки.

Выводы

Были классифицированы популярные средства разработки мобильных приложений, выполнено сравнение нативных и ненативных (кросс-платформенных) решений как по техническим характеристикам, так и по соответствию бизнес-требованиям компаний, занимающихся разработкой приложений. Приведенные рекомендации по выбору инструмента разработки могут быть полезны разработчикам, заинтересованным в выборе инструментов разработки, которые помогут быстро создавать современный программный код с учетом структуры, области применения и других факторов, которые обязательно следует учитывать при разработке программного приложения.

Список литературы

1. Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021. Statista. [Электронный ресурс]. URL: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ (дата обращения: 11.11.2022).

2. Eisenman B. Learning React Native: Building Native Mobile Apps with JavaScript. 1st ed. Sebastopol, California, USA: O'Reilly Media, 2016. 272 p.

3. Peppers J. Xamarin Cross-platform Application Development. 1st ed. г. Birmingham, United Kingdom of Great Britain: Packt Publishing, 2015. 298 p.

4. Branstein M. NativeScript in Action. 1st ed. Shelter Island, New York, USA: Manning Publications, 2017. 350 p.

5. Biessek A. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart. 1st ed. Birmingham, United Kingdom of Great Britain: Packt Publishing, 2021. 370 p.

6. Touil M. Mobile frameworks: Android Ionic and Cordova. 2st ed. Independently published, 2019. 153 p.

7. John M. Wargo PhoneGap Essentials: Building Cross-platform Mobile Apps. 1st ed. Boston, USA: AddisonWesley Professional, 2012. 384 p.

8. Develop // Apple Developer. [Электронный ресурс]. URL: https://developer.apple.com/ (дата обращения: 02.11.2022).

9. Modern Android Development // Android Developers. [Электронный ресурс]. URL: https://developer.android.com/ (дата обращения: 02.11.2022).