**Анализ методов разработки кроссплатформенных мобильных приложений**

***Чепракова Марина Алексеевна***

*студент, кафедры «Инструментального и прикладного программного обеспечения» / специальности «Программная инженерия», МИРЭА – Российского технологического университета, г. Москва, Россия*

**Аннотация**

Чепракова М.А.Анализ методов разработки кроссплатформенных мобильных приложений **/ Научно-исследовательская работа** направления профессиональной подготовки бакалавриата 09.03.04 «Программная инженерия» (7ой семестр) / кафедра ИППО Института ИТ МИРЭА – с. 18, табл. 1, ист. 69.

Каждый современный бизнес заинтересован в разработке собственного мобильного сервиса для привлечения большего количества клиентов и увеличения доходов. Таргетинг лишь на одну из двух ведущих ОС (Android и IOS) не является лучшим решением для бизнеса. Создание отдельных нативных приложений требует привлечения нескольких команд разработчиков, реализующих, по сути, одно и то же, и больше временных ресурсов. Cократить расходы и время на разработку сервиса помогают кроссплатформенные технологии. Однако необходимо четко понимать в каких проектах такие технологии принесут выгоду, а в каких наоборот убытки, и какую именно технологию выбрать для разный по назначению и сложности приложений.

В рамках работы изучен современный рынок мобильных приложений, выявлены причины появления кроссплатформенного подхода в разработке, проведено сравнение нативного и кроссплатформенного подходов к разработке мобильных приложений и определено для каких проектов выгоден каждый их них, проведен обзор наиболее популярных в настоящее время фреймворков для создания кроссплатформенных приложений, изучены направления развития кроссплатформенной разработки.

**Ключевые слова:** кроссплатформенные приложения, кроссплатформенная разработка, нативная разработка, React Native, Flutter, Kotlin Multiplatform.

**Список сокращений, обозначений, определений**

ОС Операционная Система

UI User Interface

UX User eXperience

HTTPS HyperText Transfer Protocol Secure

AOT Ahead-Of-Time

JIT Just-In-Time

PR Pull Request

IDE Integrated Development Environment

ООП Объектно-Ориентированное Программирование

MVP Minimum Viable Product

API Application Programming Interface

# **1.** **Анализ методов разработки кроссплатформенных мобильных приложений**

В число задач данного отчета по НИР входит оценка тенденций рынка мобильных приложений, сравнительный анализ нативной и кроссплатформенной разработки, обзор наиболее востребованных кроссплатформенных фреймворков.

# **1.1 Обзор современного рынка мобильных приложений и выявление предпосылок к возникновению кроссплатформенного подхода**

В настоящее время наблюдается ежедневный прирост количества пользователей смартфонов во всем мире. Число пользователей в 2020г. уже составляет 3,5 млрд. человек [1]. И, как ожидается, эта цифра будет только расти и к 2021г. достигнет отметки в 3,8 млрд. [1].

По данным отчета DATAREPORTAL мобильные приложения генерируют примерно 50% интернет-трафика [2], они предоставляют более гибкий интерфейс и широкий спектр возможностей по сравнению с мобильными версиями сайтов, за счет чего пользователи смартфонов до 91% своего экранного времени тратят именно на приложения [2].

По состоянию на 2019г. мировые доходы от покупок приложений в App Store составили $120 млрд, что в 2,1 раза больше, чем в 2016г. [16]. Доходы от мобильной рекламы согласно отчету App Annie выросли с 2019г. по 2020г. на 26%, а это $240млрд [19]. В связи с чем очевидно, что каждый современный бизнес заинтересован в разработке собственного мобильного сервиса для привлечения большего количества клиентов и увеличения доходов.

Отчет Statcounter показывает, что Android [6] является ведущей операционной системой для смартфонов в мире с долей рынка более 71% [3], тогда как доля IOS [5] составляет около 28% [3].

Учитывая приведенные выше статистические данные, можно с уверенностью сказать, что таргетинг лишь на одну из двух ведущих ОС не является лучшим решением для бизнеса. Android выигрывает по количеству пользователей, а IOS по их платежеспособности [6, 18, 41]. Однако, разработка одного приложения под несколько платформ может оказаться трудоемким и дорогостоящим вариантом. Создание отдельных нативных приложений требует привлечения нескольких команд разработчиков, реализующих, по сути, одно и то же, и больше временных ресурсов. Чем дольше команды работают над проектом, тем дороже, учитывая почасовую оплату каждого сотрудника [4], обойдется заказчику разработка сервиса.

Результатом стремления сократить расходы и время на разработку сервиса в условиях существования когда-то большого количества различных вариантов операционных систем на рынке (iOS, BlackBerry, Android, Symbian и Windows Mobile), и при этом обойтись без потерь в охвате пользователей, стало развитие кроссплатформенного метода разработки мобильных приложений, и уже сегодня он составляет реальную конкуренцию нативному, предоставляя индустрии способ разрабатывать для iOS и Android одно приложение вместо двух, охватывая сразу 99% пользователей смартфонов.

# **1.2 Сравнительный анализ нативного и кроссплатформенного подхода к созданию мобильных приложений**

# **1.2.1 Нативный подход**

Под нативным подходом подразумевается использование оригинальных языков и инструментов разработки конкретной мобильной операционной системы. Так приложения для iOS создаются в среде разработки XCode [7] на языке Swift [8], а приложения под Android в Android Studio [9] на Kotlin [10, 11] (до 2018 года основным языком был Java [12]).

Разработка мобильного приложения на родных технологиях и языках имеет следующие положительные моменты:

1. Гибкость в реализации

Можно реализовать все, на что способны технологии мобильной операционной системы. Кроме того, разработчик не ограничен в использовании сторонних библиотек.

1. Эффективность UI / UX

Поведенческие паттерны различны для, например, устройств iOS и Android. Пользователи зачастую находят реализации нативных версий более удобными.

1. Скорость работы

Полная аппаратная поддержка мобильной платформы позволяет нативным приложениям работать быстро и бесперебойно.

1. Доступность последних технологий

Новый программный и аппаратный функционал, предоставленный компаниями-производителями устройства и операционной системы, становится доступен для реализации сразу после выпуска соответствующих обновлений.

1. Легкость и качество тестирования

В процессе разработки не сложно отследить, в какой момент приложение начинает требовать большего объёма памяти или ресурсов процессора, а также обнаружить тормозящие процессы и оптимизировать их.

Более того, разработчик не ограничен в применение юнит-тестов и в интеграции систем удаленного мониторинга ошибок.

1. Полная поддержка со стороны магазинов приложений

Нативные приложения по умолчанию отвечают определенному числу требований магазинов, поэтому процесс публикации для них относительно прост.

Вместе с тем, стоит учитывать, что нативная разработка под несколько платформ требует значительных ресурсных затрат и приводит к:

1. Увеличению стоимости разработки

Для разработки сервиса придется задействовать не одну, а несколько команд разработчиков, что приведет к увеличению издержек.

Кроме того, дорожает и поддержка, после выхода на рынок необходимо обновлять функционал и исправлять ошибки сразу на нескольких платформах, при этом на каждой из них возможно появление уникальных багов, и нет гарантии, что функционал всех версий приложения будет идентичен в любой срез времени, что ведет к ухудшению пользовательского опыта на одной или сразу на нескольких платформах.

1. Увеличению времени разработки

Реализация нескольких приложений затратна по времени, кроме того, даже при одновременной работе нескольких платформенных команд стоит учитывать, что каждая из них будет иметь собственный темп разработки.

# **1.2.2 Кроссплатформенный подход**

Существует два типа кроссплатформенных мобильных приложений:

* Гибридные гибридные приложения (Cordova 2013, Ionic 3013) опираются на веб-технологии (HTML, CSS и JavaScript/ECMAScript) и выполняются обычно в WebView. По сути, это веб-сайты, имитирующие работу приложения.
* Нативные кроссплатформенные приложения (Xamarin 2011, React Native 2015, Flutter 2017, Kotlin Native 2017) пишутся на нативных или кроссплатформенных языках программирования и затем либо компилируются в ассемблерный код AOT, либо JIT, который использует нативные виджеты пользовательского интерфейса.

Выбор в пользу кроссплатформенного мобильного приложения обеспечит следующие преимущества:

1. Широкий охват рынка [17]

Расширение пользовательской базы за счёт появления приложения одновременно в нескольких магазинах.

1. Дешевизна сборки за счет снижения рабочей нагрузки [17]

Единая кодовая база позволяет инвестировать только один раз и только в одну команду.

1. Быстрая разработка сразу для нескольких платформ [17]

Единая кодовая база легче тестируется и позволяет повторно использовать до 90% кода вместо написания полностью уникального под каждую платформу. В результате можно получить преимущество быстрого выхода на рынок (TTM – Time to Market).

1. Упрощенная поддержка продукта

Разработчикам необходимо поддерживать лишь одну версию приложения. А добавление функционала и исправление багов распространяется на все платформы разом.

1. Упрощенная процедура обновления

Кроссплатформенные приложения доступны через интернет, поэтому обновляются автоматически.

1. Простая интеграция с облаком
2. Единая логика приложения на всех платформах
3. Согласованность UI и UX

При разработке кроссплатформенных приложений различия в UI/UX между Android и iOS обрабатываются по умолчанию, что делает несогласованность дизайна и навигации менее вероятной.

Основные недостатки кроссплатформенного приложения:

1. Непредсказуемость пользовательского опыта и дизайна [17]

На подсознательном уровне пользователь ждет от приложения отзывчивости. Кроссплатформенные решения обрабатывают больше данных, поскольку включают дополнительный уровень абстракции, поэтому работают медленнее нативных при наличии многоэлементного интерфейса.

Также для кроссплатформенной разработки нет гайдлайнов. У каждой из платформ свои стандарты, и если кроссплатформенное приложение разработано по гайдлайнам iOS [15], то пользователям Android [14] это доставит неудобство, и наоборот.

1. Проблемы с внедрением новых функций

Кроссплатформенные решения – стороннее ПО, поэтому они не могут обеспечить немедленную поддержку последних обновлений iOS и Android.

1. Зависимость от мобильного устройства [17]

Кроссплатформенные приложения обычно не работают оффлайн. Поэтому их возможности зависят от наличия стабильного подключения к интернету, версии операционной системы и модели устройства.

1. Проблемы с безопасностью

Для обеспечения надежной защиты требуется эффективное использование аппаратных ресурсов, с чем кроссплатформенные приложения справляются на порядок хуже нативных. А введение нового уровня поверх родной ОС приносит еще больше уязвимостей.

1. Низкие показатели производительности [17]

Любое ответное действие со стороны приложения требует выполнения процессов сериализации и десериализации.

1. Большой вес приложения [13]
2. Нет устойчивой поддержки сообщества

Сообщества разработчиков кроссплатформенных приложений в разы меньше, чем iOS- и Android-комьюнити. Поскольку экосистемы фреймворков постоянно развиваются, а обновления библиотек выходят едва ли не каждый месяц, разработчикам приходится вкладывать много времени и сил в изучение документации.

Более того, в некоторых случаях недостаток пользовательских компонентов приводит к тому, что приходится искать отдельные решения [13].

1. Сложности при построении команды

Кроссплатформенная разработка требует от разработчиков глубокого понимания целевых ОС, ведь не смотря на одну кодовую базу, им придется разворачивать приложения и в Google Play и в App Store, а также писать дополнительные куски кода под каждую платформу и думать о том, как наиболее эффективно обработать различия операционных систем.

1. Ограничения в поддержке магазинов

Кроссплатформенные приложения зачастую не соответствуют всем требованиям магазинов или же имеют по некоторым пунктам низкие показатели. В результате чего, они редко попадают в Featured-разделы App Store и Google Play.

Резюмируя вышеописанное, можно сделать вывод, что нативная разработка выгодна в том случае, если заказчику достаточно охватить аудиторию лишь одной платформы или необходимо добиться безупречного пользовательского опыта от конкретной платформы. Также в сторону нативного приложения склоняют требования к беспрепятственному использованию аппаратного обеспечения мобильного устройства или специфичных для платформы функций и высокой производительности. Кроме того, если планируется создать сложное и долгосрочное приложение с широким функционалом, многоэлементным интерфейсом или сложной анимацией, нативное приложение – единственный выбор. Об этом говорит опыт таких крупных компаний как Facebook, Airbnb которые разработали кроссплатформенные приложения для своих целей, но в итоге вернулись к нативному способу разработки, так как кроссплатформенное приложение не удовлетворяло их требованиям, с трудом масштабировалось и не оправдывало ожидания пользователей [20, 21, 23].

Однако мир программного обеспечения сильно изменился с тех пор, как Цукерберг сделал свое заявление о частичном уходе от кроссплатформенных технологий в 2012 году, когда использовался, кстати, в основном гибридный подход. Современные фреймворки уходят от гибридного подхода и предлагают собственные инструменты разработки и компиляцию в Swift и Kotlin, что позволяет повысить производительность. Каждый фреймворк имеет свои проблемные точки, но многие копании все равно продолжают инвестировать в них, за счет того, что общая прибыль с приложения превышает затраты на обход препятствий к лучшей производительности, а некоторые даже вносят свой вклад в кодовую базу фреймворков на основе опыта самостоятельных исправлений критических моментов [32].

Есть случаи, когда лишь некоторый функционал приложения разрабатывается на кроссплатформенных технологиях, таким образом подход используется максимально эффективно там, где он действительно нужен [39].

Разработаны бренд-ориентированный, платформа-ориентированный и смешанный подходы к адаптации с UI/UX в кроссплатформенных приложениях [34, 35, 36, 37], а Material Design даже выпустили best practices по этой теме [38]. Таким образом, кроссплатформенное приложение подойдет и тем, кто считает, что приложение должно выглядеть одинаково везде, и тем, кто считает, что приложение должно соответствовать платформе.

В последние годы и Google, и Apple позаимствовали декларативную систему пользовательского интерфейса из Интернета, используя ее во фреймворках Flutter, Jetpack Compose и SwiftUI. Это означает, что в будущем разработчику Swift больше не нужно понимать структуру Android XML, а разработчику Android больше не нужно знать, как работать с файлами XIB или раскадровками.

Кроссплатформенный подход позволяет сэкономить 10-30% бюджета, многие известные приложения такие как Walmart [26], Tesla [25], UberEats [24], Google Ads [27], Alibaba [28], Wix [32], Discord [33], воспользовались этим и сумели быстро выйти на рынок и получить широкий охват пользователей. Рынок, сообщество и популярность кроссплатформенных фреймворков растет, о чем говорят статистические данные Markets And Markets research [31], Google Trends [30] и Stack Overflow Trends [29]. Все вышеперечисленное говорит о том, что кроссплатформенный метод разработки – это хорошая практика, которая значительно упростит реализацию приложения сразу под несколько платформ, но подходит она не под каждую задачу, нужно обязательно анализировать тип приложения, стек компании и опыт команды разработчиков. Выгодно использовать данный подход при построении проектов автоматизации бизнес-процессов, B2B-приложений с массивной бизнес-логикой, приложений для распространения контента, игровых [42] и сравнительно простых в своем функционале, интерфейсе и анимациях приложений.

# **1.3 Обзор фреймворков для кроссплатформенной разработки**

В рамках данного раздела будут рассмотрены нативные кроссплатформенные фреймворки: React Native, Flutter и Kotlin Multiplatform, так как они занимают большую часть кроссплатформенного рынка и их популярность все еще растет, в отличии от Xamarin [49], вместе с тем, как прогрессивные (PWA) и гибридные фреймворки (Cordova, Sencha, Ionic) уже реже используются в разработке [49] из-за низкой производительности, неточного интерфейса и наличия JavaScript моста. Основная информация о рассматриваемых фреймворках приведена в таблице 1.

***Таблица 1.***

**Основная информация**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **React Native** | **Flutter** | **Kotlin Multiplatform** |
| Разработчик и год первого релиза | Facebook + сообщество 2015 | Google + сообщество 2017 | JetBrains 2017 |
| Язык | JavaScript + React | Dart | Kotlin, Swift, Java |
| Платформы | Android, iOS, Windows, MacOS | Android, iOS, Google, Fuchsia, Web, Desktop | Android, iOS, watchOS, tvOS, macOS, Linux, Windows |
| Открытый ресурс | да | да | да |
| Производительность | Высокая | Очень высокая | Очень высокая |
| Горячая перезагрузка | да | да | да |
| Время разработки | Малое | Малое | Малое |
| Стоимость разработки | Средняя | Высокая | Высокая |
| Переиспользование кода | До 90% | До 90% | 75-85% (только логика) |

# **1.3.1 React Native**

Данный фреймворк - продукт внутреннего хакатона, проведенного в 2013 году Facebook. Компания хотела перенести на мобильные устройства преимущества веб-разработки, такие как быстрые итерации и наличие единой команды для создания всего продукта.

GitHub - 3,7 тыс. участников, 92,2 тыс. звезд, 299 PR [43].

Stack Overflow: один из самых «страшных» фреймворков [46].

**Достоинства:**

1. Регулярные обновления и реакция на потребности разработчиков. Растущее признание сообщества, простота поиска разработчиков.
2. Потенциал RN выходит за рамки разработки мобильных приложений [44, 63]. Фреймворк поддерживают Microsoft и Shopify [44, 45].
3. Поддерживается большинством IDE.
4. Использует самый популярный язык JavaScript
5. Много готовых компонентов и сторонних библиотек.
6. Хорошая совместимость со сторонними плагинами (Google Maps).
7. Почти нативный пользовательский опыт благодаря использованию контроллеров, встроенных как в Android, так и в iOS. Использование библиотеки ReactJS с элементами пользовательского интерфейса, что позволяет ускорить процесс проектирования.
8. Доступ к акселерометру и камере, однако для использования этих функций необходимо писать нативный код.

**Недостатки:**

1. Готовые компоненты зачастую требуют дополнительных корректировок в коде.
2. Продолжительный процесс тестирования и отладки.
3. Не является полностью кроссплатформенным, для реализации многих функций необходимо писать нативный код.
4. Наличие JavaScript моста

Используется в приложениях: Tesla, Walmart, Pulse SoundCloud, Bloomberg, Pinterest, Instagram.

Опыт использования: [50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 65].

# **1.3.2 Flutter**

Данный фреймворк - продукт Google [64], его принцип – создание приложений с единой кодовой базой для мобильных платформ, веба и десктопа.

GitHub - 3,3 тыс. участников, 109 тыс. звезд, 171 PR [47].

Stack Overflow: один из самых «любимых» фреймворков [46].

**Достоинства:**

1. Использует оптимизированный язык Dart, который имеет современные подходы, многопарадигмальность и ООП для создания кроссплатформенных приложений. Dart считается легким в освоении по причине того, что заимствован из C/C++ и Java и имеет обширную и понятную документацию.
2. Приложения запускаются с помощью изначально скомпилированного кода.
3. Высокая производительность, так как собственные компиляторы, набор виджетов и графический движок Skia устраняют необходимость в мосте JavaScript. Рассчитан на достижение постоянной скорости 60 кадров в секунду.
4. Возможность создавать сложные дизайны пользовательского интерфейса с богатыми переходами и анимацией
5. Создан владельцами Android, поддерживается Google. Удобен для Android-разработчиков.
6. Поддерживается Android Studio / IntelliJ и Visual Studio Code.
7. Поддерживает API Firebase и другие сервисы Google.
8. Flutter rendering engine предлагает большой набор готовых настраиваемых виджетов [66]. Следует подходу «UI-as-a-code». Позволяет создать приятный, продуманный, пиксельный интерфейс с использованием как собственных, не привязанных к платформе, так и имитирующих привычные iOS/Android элементов. Поддерживает material design.
9. Позволяет выпускать приложения для разных версий Android и iOS без дополнительных корректировок в коде.
10. Идеально подходит для создания MVP приложений.
11. Flutter – это реактивный фреймворк.
12. Облегченное и быстрое по времени тестирование при отсутствии специфичных для платформ функций.

**Недостатки:**

1. Необходимо изучать новый язык.
2. Нет автоматической адаптации стилей для разных платформ.
3. По причине незрелости инструмента может возникнуть необходимость в написании собственных функциональных элементов. Мало поддерживаемых библиотек и плагинов. Не поддерживает сервисы Amazon Web Services.
4. Не позволяет полностью абстрагироваться от платформы, требуются нативные знания и неизбежна концепция трех кодовых баз при использовании специфичных для платформ функций.
5. Недостаточно развитое сообщество.
6. Возможны проблемы с публикацией в магазинах приложений [64].
7. Приложения не поддерживаются веб-браузерами.
8. Большой размер приложения.
9. Не предлагает поддержки Android TV и Apple TV.

Используется в приложениях: Alibaba, Hamilton Musical, Greentea, Google Ads, Philips Hue.

Опыт использования: [60, 61, 62].

# **1.3.3 Kotlin Multiplatform**

У JetBrains есть важное отличие от многих других сторонников кроссплатформенности: там изначально не ставят целью «полностью универсальную кодовую базу». Радикальный подход «всё общее» в большинстве случаев работает плохо или накладывает массу ограничений, поэтому компания и не пытается его воплотить.

Вместо этого предлагают переиспользовать только по-настоящему общую часть кода. А всё различающееся писать нативно. UI здесь никак не объединяют.

Всё общее можно выделить в библиотеку на Kotlin, подходящую сразу для iOS и Android, а дальше обращаться к ней с обеих сторон.

Выбрав подход «проект из трёх частей», JetBrains направили усилия на максимально бесшовное взаимодействие этих частей на уровне языка.

GitHub – 1,3 тыс. участников, 34,3 тыс. звезд, 171 PR [47].

Stack Overflow: Kotlin один из самых «интересующих» языков [46].

**Достоинства:**

1. Благодаря популярности Kotlin в Android-разработке «три кодовых базы» могут писаться на одном языке. Многие мобильные разработчики уже пишут на Kotlin. iOS-разработчикам возможно придётся изучить новый язык, но этот процесс упростит сходство Kotlin со Swift.
2. Kotlin позволяет создавать максимально чистый код.
3. Облегченное и быстрое тестирование
4. Команды нативных программистов могут распределять задачи между собой и использовать код друг друга в дальнейшем.
5. UI/UX разрабатываются отдельно для каждой платформы, хороший пользовательский опыт.

**Недостатки:**

1. Границы платформ размываются и появляется «общая» часть, но она не одинаково близка обеим сторонам.
2. Очень маленькое сообщество. Сложно найти разработчиков, экспертов в технологии.
3. Возможно, придется изучать новые язык программирования и среду разработки.
4. В команде обязательно наличие нативных разработчиков.

Используется в приложениях: Яндекс.Карты и Яндекс.Диск (несколько модулей), Netflix, Quizlet, Autodesk.

Опыт использования: [48, 67, 68, 69].

# **Заключение**

Нет единственно верного решения при выборе подхода к мобильной разработке. Однако, за кроссплатформенными приложениями будущее, они предлагают более продвинутые функции, чем нативный инструментарий и помогают экономить.

React Native следует выбрать, если:

* Команда разработчиков знакома с веб-разработкой и JavaScript.
* Осуществляется перенос веб-приложения на iOS и Android.

Ftlutter следует выбрать, если:

* Планируется MVP-приложение
* Есть необходимость в качественной анимации
* Есть необходимость в высокой производительности
* Нет сильной необходимости в специфичных функциях платформ и доступе к аппаратному обеспечению.

Подход Kotlin Multiplatform выглядит наиболее продуманным и эффективным, скорее всего именно на нем будут строиться кроссплатформенные приложение в будущем. Совместное использование лишь бизнес-логики на разных платформах приносит стоящие выгоды, кроме того, все приложение целиком может писаться на одном уже знакомом мобильным разработчикам языке. Это новый инструмент, а не замена существующего набора инструментов.

# **Список источников**

1. Smartphone users worldwide 2020 / Statista [Электронный ресурс]. — URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (Дата обращения 01.12.2020).
2. DIGITAL 2020: GLOBAL DIGITAL OVERVIEW [Электронный ресурс]. — URL: https://datareportal.com/reports/digital-2020-global-digital-overview (Дата обращения 01.12.2020).
3. Mobile Operating System Market Share Worldwide / Statcounter [Электронный ресурс]. — URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (Дата обращения 01.12.2020).
4. How much does it cost to make an app? [Электронный ресурс]. — URL: https://thinkmobiles.com/blog/how-much-cost-make-app/ (Дата обращения 01.12.2020).
5. iOS 14 Apple [Электронный ресурс]. — URL: https://www.apple.com/ru/ios/ios-14/ (Дата обращения 01.12.2020).
6. Android 11 [Электронный ресурс]. — URL: https://www.android.com/ (Дата обращения 01.12.2020).
7. Xcode 12 [Электронный ресурс]. — URL: https://developer.apple.com/xcode/ (Дата обращения 01.12.2020).
8. Swift – Apple [Электронный ресурс]. — URL: https://www.apple.com/ru/swift/ (Дата обращения 01.12.2020).
9. Android Studio [Электронный ресурс]. — URL: https://developer.android.com/studio?hl=ru (Дата обращения 01.12.2020).
10. Kotlin [Электронный ресурс]. — URL: https://kotlinlang.org/ (Дата обращения 01.12.2020).
11. Develop Android apps with Kotlin [Электронный ресурс]. — URL: https://developer.android.com/kotlin?hl=ru (Дата обращения 01.12.2020).
12. Oracle Java [Электронный ресурс]. — URL: https://www.oracle.com/ru/java/ (Дата обращения 01.12.2020).
13. Нативная или кроссплатформенная разработка? Наглядное сравнение. [Электронный ресурс]. — URL: https://appcraft.pro/blog/pochemu-my-vybiraem-nativnuyu-razrabotku/ (Дата обращения 01.12.2020).
14. Material Design [Электронный ресурс]. — URL: https://material.io/design (Дата обращения 01.12.2020).
15. Human Interface Guidelines [Электронный ресурс]. — URL: https://developer.apple.com/design/human-interface-guidelines/ (Дата обращения 01.12.2020).
16. App Annie State of Mobile in 2020 Report: 10 Highlights [Электронный ресурс]. — URL: https://medium.com/@nadir/app-annie-state-of-mobile-in-2020-report-10-highlights-7746d7aad044 (Дата обращения 01.12.2020).
17. React Native: A retrospective from the mobile-engineering team at Udacity [Электронный ресурс]. — URL: https://engineering.udacity.com/react-native-a-retrospective-from-the-mobile-engineering-team-at-udacity-89975d6a8102 (Дата обращения 01.12.2020).
18. App Revenue Statistics (2019) [Электронный ресурс]. — URL: https://www.businessofapps.com/data/app-revenues/ (Дата обращения 01.12.2020).
19. REPORT The State of Mobile 2020 [Электронный ресурс]. — URL: https://www.appannie.com/ru/go/state-of-mobile-2020/ (Дата обращения 01.12.2020).
20. Under the hood: Rebuilding Facebook for iOS [Электронный ресурс]. — URL: https://www.facebook.com/notes/facebook-engineering/under-the-hood-rebuilding-facebook-for-ios/10151036091753920/ (Дата обращения 01.12.2020).
21. Facebook’s Zuckerberg [Электронный ресурс]. — URL: https://venturebeat.com/2012/09/11/facebooks-zuckerberg-the-biggest-mistake-weve-made-as-a-company-is-betting-on-html5-over-native/ (Дата обращения 01.12.2020).
22. How Much Does It Cost to Make an App in 2021? [Электронный ресурс]. — URL: https://www.velvetech.com/blog/how-much-mobile-app-cost/#1\_CrossPlatform\_Development (Дата обращения 01.12.2020).
23. Why Airbnb is Moving Off Of React Native [Электронный ресурс]. — URL: https://softwareengineeringdaily.com/2018/09/24/show-summary-react-native-at-airbnb/ (Дата обращения 01.12.2020).
24. Powering UberEATS with React Native and Uber Engineering [Электронный ресурс]. — URL: https://eng.uber.com/ubereats-react-native/ (Дата обращения 01.12.2020).
25. Tesla App Built In React Native [Электронный ресурс]. — URL: https://www.reddit.com/r/reactnative/comments/5iv7mi/tesla\_app\_built\_in\_react\_native/ (Дата обращения 01.12.2020).
26. A new beginning for React Native at WalmartLabs Online Grocery [Электронный ресурс]. — URL: https://medium.com/walmartglobaltech/a-new-beginning-for-react-native-at-walmartlabs-online-grocery-c30b27db57d3 (Дата обращения 01.12.2020).
27. Google native ads integrated into a Flutter widget tree [Электронный ресурс]. — URL: https://github.com/flutter/flutter/issues/12114 (Дата обращения 01.12.2020).
28. Alibaba used Flutter to build 50+ million user Xianyu app [Электронный ресурс]. — URL: https://www.reddit.com/r/FlutterDev/comments/9h8ycp/alibaba\_used\_flutter\_to\_build\_50\_million\_user/ (Дата обращения 01.12.2020).
29. Stack Overflow Trends [Электронный ресурс]. — URL: https://insights.stackoverflow.com/trends?tags=react-native%2Cflutter%2Cxamarin%2Cionic%2Ckotlin%2Ccordova (Дата обращения 01.12.2020).
30. Google Trends [Электронный ресурс]. — URL: https://trends.google.ru/trends/explore?geo=RU&q=react%20native,flutter,xamarin,kotlin%20native,Ionic (Дата обращения 01.12.2020).
31. Cross-Platform and Mobile Advertising Market [Электронный ресурс]. — URL: https://www.marketsandmarkets.com/PressReleases/mobile-advertising.asp (Дата обращения 01.12.2020).
32. React Native at Wix [Электронный ресурс]. — URL: https://medium.com/wix-engineering/react-native-at-wix-d56c3745aaff (Дата обращения 01.12.2020).
33. Why Discord is Sticking with React Native [Электронный ресурс]. — URL: https://blog.discord.com/why-discord-is-sticking-with-react-native-ccc34be0d427 (Дата обращения 01.12.2020).
34. Avoiding Bad Practices in iOS and Android Design [Электронный ресурс]. — URL: https://www.toptal.com/android/android-and-ios-app-design-tips (Дата обращения 01.12.2020).
35. Approaches For Multiplatform UI Design Adaptation [Электронный ресурс]. — URL: https://www.smashingmagazine.com/2015/09/approaches-for-multiplatform-ui-design-adaptation/ (Дата обращения 01.12.2020).
36. How to Design UI for Cross Platform Devices [Электронный ресурс]. — URL: https://medium.com/hitech-nudge-rewards/how-to-design-ui-for-cross-platform-devices-5809e008935d (Дата обращения 01.12.2020).
37. Cross-platform app design: 3 ways to build React Native UI/UX [Электронный ресурс]. — URL: https://www.purrweb.com/blog/cross-platform-design/ (Дата обращения 01.12.2020).
38. Cross-platform adaptation [Электронный ресурс]. — URL: https://material.io/design/platform-guidance/cross-platform-adaptation.html (Дата обращения 01.12.2020).
39. Натив или гибрид? Специалисты Яндекса отвечают на главный вопрос мобильной разработки [Электронный ресурс]. — URL: https://habr.com/en/company/yandex/blog/326882/ (Дата обращения 01.12.2020).
40. React Native vs Swift - Performance and Development Comparison [Электронный ресурс]. — URL: https://www.netguru.com/blog/swift-vs-react-native (Дата обращения 01.12.2020).
41. Global App Revenue Reached [Электронный ресурс]. — URL: https://sensortower.com/blog/app-revenue-and-downloads-1h-2019 (Дата обращения 01.12.2020).
42. App Annie State of Mobile in 2020 Report [Электронный ресурс]. — URL: https://medium.com/@nadir/app-annie-state-of-mobile-in-2020-report-10-highlights-7746d7aad044 (Дата обращения 01.12.2020).
43. Facebook react-native / Github [Электронный ресурс]. — URL: https://github.com/facebook/react-native (Дата обращения 01.12.2020).
44. Microsoft react-native-windows / Github [Электронный ресурс]. — URL: https://github.com/Microsoft/react-native-windows (Дата обращения 01.12.2020).
45. React Native is the Future of Mobile at Shopify [Электронный ресурс]. — URL: https://shopify.engineering/react-native-future-mobile-shopify (Дата обращения 01.12.2020).
46. Stack Overflow’s Developer Survey 2020 [Электронный ресурс]. — URL: https://insights.stackoverflow.com/survey/2020#overview (Дата обращения 01.12.2020).
47. flutter / Github [Электронный ресурс]. — URL: https://github.com/flutter/flutter (Дата обращения 01.12.2020).
48. Опыт работы с Kotlin Multiplatform за 10 месяцев [Электронный ресурс]. — URL: https://medium.com/icerock/опыт-работы-с-kotlin-multiplatform-за-10-месяцев-435a7e08e52d (Дата обращения 01.12.2020).
49. Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020 [Электронный ресурс]. — URL: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ (Дата обращения 01.12.2020).
50. Why Airbnb is Moving Off Of React Native [Электронный ресурс]. — URL: https://softwareengineeringdaily.com/2018/09/24/show-summary-react-native-at-airbnb/ (Дата обращения 01.12.2020).
51. 13 Great Examples of React Native Apps in 2020 [Электронный ресурс]. — URL: https://www.netguru.com/blog/react-native-apps (Дата обращения 01.12.2020).
52. React Native vs Swift - Performance and Development Comparison [Электронный ресурс]. — URL: https://www.netguru.com/blog/swift-vs-react-native (Дата обращения 01.12.2020).
53. React Native Pros and Cons - Facebook’s Framework in 2020 [Электронный ресурс]. — URL: https://www.netguru.com/blog/react-native-pros-and-cons (Дата обращения 01.12.2020).
54. React Native at Airbnb [Электронный ресурс]. — URL: https://medium.com/airbnb-engineering/react-native-at-airbnb-f95aa460be1c (Дата обращения 01.12.2020).
55. React Native is the Future of Mobile at Shopify [Электронный ресурс]. — URL: https://shopify.engineering/react-native-future-mobile-shopify (Дата обращения 01.12.2020).
56. React Native at Wix [Электронный ресурс]. — URL: https://medium.com/wix-engineering/react-native-at-wix-d56c3745aaff (Дата обращения 01.12.2020).
57. Why Discord is Sticking with React Native [Электронный ресурс]. — URL: https://blog.discord.com/why-discord-is-sticking-with-react-native-ccc34be0d427 (Дата обращения 01.12.2020).
58. React Native at SoundCloud [Электронный ресурс]. — URL: https://developers.soundcloud.com/blog/react-native-at-soundcloud (Дата обращения 01.12.2020).
59. Why Discord is Sticking with React Native [Электронный ресурс]. — URL: https://blog.discord.com/why-discord-is-sticking-with-react-native-ccc34be0d427 (Дата обращения 01.12.2020).
60. Top Apps Made with Flutter – 18 Stories by Developers and Business Owners [Электронный ресурс]. — URL: https://www.thedroidsonroids.com/blog/apps-made-with-flutter (Дата обращения 01.12.2020).
61. Lessons Learned from Cross-platform Mobile Development with Flutter [Электронный ресурс]. — URL: https://www.netguru.com/blog/lessons-learned-from-cross-platform-mobile-development-with-flutter (Дата обращения 01.12.2020).
62. Новое приложение «Медузы». Почему Flutter? [Электронный ресурс]. — URL: https://habr.com/ru/company/meduza/blog/501786/ (Дата обращения 01.12.2020).
63. React Native for Windows + macOS [Электронный ресурс]. — URL: https://microsoft.github.io/react-native-windows/ (Дата обращения 01.12.2020).
64. What is Flutter? [Электронный ресурс]. — URL: https://flutter.dev/docs/resources/faq#will-apple-reject-my-flutter-app (Дата обращения 01.12.2020).
65. Supporting React Native at Pinterest [Электронный ресурс]. — URL: https://medium.com/pinterest-engineering/supporting-react-native-at-pinterest-f8c2233f90e6 (Дата обращения 01.12.2020).
66. Widget catalog Flutter [Электронный ресурс]. — URL: https://flutter.dev/docs/development/ui/widgets (Дата обращения 01.12.2020).
67. Yandex / Kotlin Multiplatform Mobile [Электронный ресурс]. — URL: https://kotlinlang.org/lp/mobile/case-studies/yandex/ (Дата обращения 01.12.2020).
68. Тематическое исследование GitLive: мультиплатформенность Kotlin [Электронный ресурс]. — URL: https://blog.jetbrains.com/kotlin/2020/11/gitlive-case-study-kotlin-multiplatform-across-desktop-and-mobile/ (Дата обращения 01.12.2020).
69. Case Studies / Kotlin Multiplatform Mobile [Электронный ресурс]. — URL: https://kotlinlang.org/lp/mobile/case-studies/ (Дата обращения 01.12.2020).