Белый Марк Владимирович
студент
Тепляков Андрей Алексеевич
студент

ФГБОУ ВО «Кубанский государственный аграрный
университет им. И.Т. Трубилина»
г. Краснодар, Краснодарский край

СРАВНЕНИЕ КРОСС-ПЛАТФОРМЕННОЙ И НАТИВНОЙ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Аннотация: *авторы статьи отмечают, что в 21 в. мобильные телефоны стали неотъемлемой частью взаимодействия человека с цифровой средой. Развитие Android и iOS привело к формированию двух основных подходов к созданию мобильного программного обеспечения (ПО): нативного и кросс-платформенного. Каждый обладает собственными характеристиками, определяющими скорость разработки, производительность и качество пользовательского опыта. От выбора подхода зависят не только затраты компаний, но и то, насколько сервисы смогут соответствовать требованиям пользователя. Цель данной статьи – исследовать нативную и кросс-платформенную разработки, выявить их преимущества и недостатки, а также оценить влияние выбранного подхода на долгосрочную поддержку мобильных приложений.*

Ключевые слова: *кросс-платформенная разработка, нативная разработка, мобильные приложения, производительность, пользовательский опыт, мобильные платформы, Android, iOS, фреймворки.*

За последние годы мир программного обеспечения для смартфонов претерпел сильные изменения. Появились новые технологии, повлиявшие на формирование различных подходов к разработкам программ. Основными стали: нативная, использующая инструменты и языки представителями платформ, и кросс-платформенная, которая предоставляет один код под различные мобильные системы.

В данной статье мы проанализируем различия нативной кросс-платформенной разработок, выделяя их положительные и отрицательные моменты.

Нативная разработка предполагает собой создание продукта под конкретную операционную систему. Если последний нацелен на несколько систем, то его разработка будет проводиться отдельно, на разных языках.

Кросс-платформенная же не требует написания кода раздельно для каждой операционной системы. Она строится таким образом, чтобы один и тот же код работал во всех системах [1].

Основными инструментами для кросс-платформенной мобильной разработки являются: React native и Flutter. React native – фреймворк, созданный компанией Facebook в 2015 г. проект с открытым исходным кодом «open source». Flutter – фреймворк создала компания Google в 2017 г. как аналогичный информационный продукт [2].

Кросс-платформенная разработка предполагает написание единого кода под различные операционные системы, такие как IOS и Android, что подразумевает использование одного и того же User Interface(UI), вследствие чего пользователи не заметят никакой разницы. Посредством нее можно быстрее разработать приложение. Но, в отличие от нативной, его действие будет замедленно и не гарантируется пользовательская безопасность. Тем не менее есть риски, что кросс-платформенный фреймворк может в некоторых случаях не иметь полного доступа к какой-либо системе (не иметь всего функционала устройства) или некорректно работать. Еще одним недостатком кросс-платформенной разработки являются свежие обновления Операционных Систем (ОС) Android и IOS, которые могут не так быстро появляться в фреймворках, как это происходит при нативной. Если в приложении много логики (алгоритмов, правил, управляющих конструкций и т. д.) и есть необходимость сделать ее многопоточной, то это может в последующем стать весомой проблемой. Также кросс-платформенные фреймворки имеют достаточно тяжелую исполнительную среду, что делает их более затратными к процессору и оперативной памяти. Стоит учитывать и то, что

лучше использовать нейтральный UI, чтобы не создавать потенциальных проблем с различным поведением на платформах. Исходя из представленной выше информации сформируем положительные и отрицательные моменты кросс-платформенной разработки [3].

К негативным относятся: увеличенное энергопотребление; меньшая производительность; сложное поддерживание функционала, привязанного к конкретному устройству; большая вероятность ошибок внутри кросс-платформенных библиотек; не покрывает все кейсы разработки; ограничения по дизайну.

Позитивные выражаются в следующем: ускоренная реализация MVP; меньшая стоимость разработки; единая кодовая база; более быстрая разработка; меньше трудозатрат и сниженная вероятность ошибок в приложениях из-за более легкого тестирования.

Кросс-платформенной разработки применимы в стартапах и MVP, электронной коммерции, медиа и контент сервисах, приложениях малого и среднего бизнеса, туризме и путешествиях. Что, в свою очередь, требует соблюдения информационной безопасности.

Если приложение будет в полной мере использовать все ресурсы устройства: как программные, так и аппаратные, то рациональным решением будет остановить свой выбор на нативной разработке, поскольку большинство крупных проектов и стартапов – именно нативная разработка. Нередко бывает так, что продукт выходит сначала на Android, а через некоторое время и на IOS [3; 4].

Для разработки на базе последней операционной системы используется одна среда – XCODE. Язык, на котором пишется приложение – Swift, выпущен в 2014 г. компанией Apple. XCODE является единственной интегрированной средой разработки (IDE), а использовать ее можно исключительно на компьютерах и ноутбуках фирмы Apple [5].

Для разработки на Android применяется больше решений, чем на IOS: Android studio, Intellij IDEA, Eclipse. Доступные языки: Java и Kotlin [6].

Android studio – самая часто используемая, мощная, фундаментальная интегрированная среда разработки (IDE) для разработки приложений под Android,

выпущенная компанией Google в 2014 г. Kotlin – основной язык разработки от Jetbrains (2011) [6].

К отрицательным чертам нативной разработки относятся: необходимость создания двух отдельных приложений; работа занимает на 30–40% больше времени; удорожание проекта (необходимо привлечь как минимум двух отдельных специалистов для Android и IOS).

Положительные моменты сводятся к следующему: полная адаптация под конкретную операционную систему при использовании всех ее возможностей; указанные приложения обеспечивают высокую производительность и удобство применения на практике; повышенная защита (приложения проходят проверку на соответствие стандартам безопасности операционной системы); строгий контроль над процессом разработки и настройка конечного приложения; как правило, пользовательский опыт в таких приложениях лучше.

Отличительные особенности производительности нативных фреймворков и кросс-платформенных представлены в таблице [3].

Таблица 1

Сравнительная характеристика производительности нативных и кросс-платформенных фреймворков

Фреймворк	Стартап-время	Частота кадров	Размер приложения
Jetpack compose/SwiftUI	Базовое значение	60 fps	Минимальный (оптимальный)
Flutter	Немного выше нативного	55–60 fps	Значительно больше нативного
React native	Значительно выше нативного	50–55 fps	Переменный

Где стартап время – это период, когда пользователь видит первый отрисованный контент, а приложение перестает быть пустым экраном; частота кадров – это показатель стабительности рендеринга при скроллинге и анимации; размер приложения – конечный его вес.

Примерный ориентир трудозатрат на одно и то же приложение:

− Flutter – 8 недель;

− React native – 9 недель;

− Jetpack Compose + SwiftUI (нативные) – 12 недель.

Таким образом, нативная разработка обладает достаточным количеством преимуществ по сравнению с кросс-платформенной. Она позволяет поддерживать высокую производительность, а также соблюдать лучшие практики разработки интерфейсов для двух платформ, однако есть сферы, в которых кроссплатформенные решения являются наиболее оптимальными по соотношению цены и качества.

Стремительно развивающиеся информационные технологии существенно влияют как на экономическую и политическую обстановку в мире, формируя массу вызовов, которые требуют своевременного ответа [7; 8].

Список литературы

1. Веденеев И. Мобильная разработка: Cross-platform или Native / И. Веденеев // Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/agima/articles/586092/ (дата обращения: 02.12.2025).

2. Роуз Р. Flutter и Dart: сборник рецептов: разработка полнофункциональных облачных приложений / Р. Роуз; пер. с англ. – Астана: Алист, 2024. – 279 с.

3. Ворожищев А. Выбираем между кросс-платформенной и нативной разработкой / А. Ворожищев // Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/agima/articles/736984/ (дата обращения: 02.12.2025).

4. Ибатулин Н. Нативная разработка vs кросс-платформенная – нужно ли выбирать? / Н. Ибатулин // Habr [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/505482/ (дата обращения: 02.12.2025).

5. Казанский А.А. Разработка приложений на Swift и SwiftUI с нуля / А.А. Казанский. – 2-е изд., перераб. – СПб.: БХВ-Петербург, 2022. – 416 с.

6. Попков Д.С. Разработка Android приложений с Jetpack Compose / Д.С. Попков. – SelfPub, 2022. – 113 с.

7. Российские информационные вызовы и ответы на них / Г.В. Арустамян, Д.О. Шестак, Р.А. Дилбандян [и др.] // Культура Мира. – 2025. – Т. 13. №47(4). – С. 181–195. EDN IKDNZR

8. Российские ответы на технологические вызовы: стратегии и решения в условиях современного менеджмента / О.В. Терещенко, Д.В. Еськов, С.А. Шульга [и др.] // Естественно-гуманитарные исследования. – 2025. – №4(60). – С. 842–847. EDN PFQHPU