ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК 004+45
DOI: 10.21822/2073-6185-2025-52-1-122-133

Обзорная статья/ Review article

Сравнительный анализ фреймворков для разработки мобильных приложений: нативные, гибридные или кроссплатформенные решения

А.Б. Темирова
Грозненский государственный нефтяной технический университет
имени акад. М.Д.Миллионщикова,
364051, г. Грозный, пр. Исаева,100, Россия

Резюме. Цель. Целью исследования является проведение сравнительного анализа различных фреймворков для разработки мобильных приложений: нативных, гибридных и кроссплатформенных решений. **Метод.** Для достижения цели использовались методы анализа, синтеза и сравнения. Проанализированы характеристики различных фреймворков для разработки мобильных приложений, включая их производительность, стоимость и доступ к возможностям устройств. **Результат**. Выявлено, что нативные фреймворки отличаются наивысшей производительностью и способностью обеспечить максимально нативный вид и функциональность приложения. Однако такой подход имеет ограничения, поскольку требует отдельной разработки для каждой платформы, что приводит к увеличению временных и ресурсных затрат. Гибридные решения оказались экономически эффективными, позволяя использовать единую кодовую базу для создания приложений для разных платформ,что упрощает процесс разработки и сопровождения. Гибридные приложения могут иметь ограниченную производительность из-за использования WebView для отображения интерфейса и ограниченного доступа к возможностям устройств. Кроссплатформенные фреймворки обеспечивают баланс между производительностью и эффективностью использования ресурсов. Они позволяют использовать единую кодовую базу для создания приложений для нескольких платформ и могут достигать удовлетворительной производительности. Однако они могут иметь ограниченный доступ к определенным возможностям устройств и внешнему виду приложений. **Вывод.** Результаты исследования вносят новый вклад в науку, предоставляя подробный сравнительный анализ различных подходов к разработке мобильных приложений и фреймворков, используемых для их создания. Полученные результаты могут быть использованы для принятия обоснованных решений относительно выбора фреймворка для разработки мобильных приложений.

Ключевые слова: Java; разработка ПО; производительность; собственный вид; кодовая база

Для цитирования: А.Б. Темирова. Сравнительный анализ фреймворков для разработки мобильных приложений: нативные, гибридные или кроссплатформенные решения. Вестник Дагестанского государственного технического университета. Технические науки. 2025; 52(1):122-133. DOI:10.21822/2073-6185-2025-52-1-122-133

Comparative analysis of frameworks for mobile application development: native, hybrid or cross-platform solutions

A.B. Temirova
M.D. Millionshchikov Grozny State Oil Technical University,
100 Isaev Ave., Grozny 364051, Russia

Abstract. Objective. The purpose of this study is to conduct a comparative analysis of various frameworks for the development of mobile applications: native, hybrid and cross-platform solutions. **Method**. To achieve this goal, methods of analysis, synthesis and comparison

were used. The characteristics of various mobile application development frameworks were analyzed, including their performance, cost, and access to device capabilities. **Result**. Native frameworks have been found to have the highest performance and the ability to provide the most native look and functionality for an application. This approach has limitations as it requires separate development for each platform, which leads to increased time and resource costs. Hybrid solutions have proven to be cost-effective, allowing you to use a single code base to create applications for different platforms. This simplifies the development and maintenance process. Hybrid apps may have limited performance due to the use of WebView to display the interface and limited access to device capabilities. Cross-platform frameworks, on the other hand, provide a balance between performance and resource efficiency. They allow you to use a single code base to create applications for multiple platforms and can achieve satisfactory performance. However, they may have limited access to certain device capabilities and the appearance of applications. **Conclusion.** The results make a new contribution to the field by providing a detailed analysis of mobile app development approaches and frameworks used to build them. The results can be used to make informed decisions regarding the choice of framework for mobile app development.

Keywords: Java; software development; performance; native view; code base

For citation: A.B. Temirova. Comparative analysis of frameworks for mobile application development: native, hybrid or cross-platform solutions. Herald of Daghestan State Technical University. Technical Sciences. 2025; 52(1):122-133. (In Russ) DOI:10.21822/2073-6185-2025-52-1-122-133

Введение. В современном информационном обществе разработка мобильных приложений стала одной из самых важных и актуальных областей разработки программного обеспечения. С каждым годом количество мобильных устройств растет, а вместе с ними и спрос на высококачественные и эффективные приложения. Выбор правильного подхода к разработке мобильных приложений является критически важной задачей для разработчиков и бизнеса. Исследуемый вопрос заключается в определении того, какой тип фреймворка для разработки мобильных приложений является наиболее эффективным и отвечает потребностям современного рынка программного обеспечения. Вопрос становится все более актуальным в связи с ростом числа доступных разработчикам фреймворков и подходов, расширяющих спектр возможностей мобильных приложений. В нескольких исследованиях были проанализированы и сравнены различные аспекты разработки мобильных приложений, включая выбор между нативным и гибридным подходами, определение лучших практик кроссплатформенной разработки и изучение методов выбора оптимальной платформы для кроссплатформенной разработки [1-7].

В работе Х.О. Козуб и Ю.Х. Козуб (Kozub, H. & Kozub, Yu. 2022) [8] исследовали разработку кроссплатформенных мобильных приложений с использованием Kotlin Multiplatform и Jetpack Compose для различных операционных систем, включая Android, Windows, Linux и macOS. Они демонстрируют важность этих инструментов и потенциал сокращения времени разработки и предотвращения ошибок благодаря их использованию. Кроме того, авторы подчеркивают необходимость использования декларативного подхода при создании пользовательских интерфейсов.

О. Каратанов и соавторы (Karatanov, O., Yena, M., Bova, Y., & Ustymenko, O., 2021) провели сравнительный анализ двух важных фреймворков для модульного тестирования на языке программирования Java – JUnit и TestNG [7]. Они определили основные функции и преимущества обоих фреймворков, отметив, что Test- NG обладает более широкими функциональными возможностями, что делает его более гибким для сложных проектов.

В исследовании М. Сингха и Г. Шобхи (Singh, M., & Shobha, G., 2021) [13] подчеркивалась актуальность разработки кроссплатформенных мобильных приложений. Были изучены различные фреймворки для этой цели и проведен сравнительный анализ их функциональности. Основные выводы исследования подчеркивают необходимость обос-

нованного выбора структуры, основанной на конкретных требованиях проекта, не существует универсальной структуры для всех ситуаций. В статье Т. Зохуда и С. Зейна (Zohud, T., & Zein, S., 2021) [20] авторы исследуют подходы команд разработчиков к разработке кроссплатформенных мобильных приложений. Они используют качественные исследования, включая метод тематических исследований, интервью и групповые дискуссии, для сбора информации от четырех компаний-разработчиков программного обеспечения в Палестине. Результаты исследования показывают, что опыт разработчиков является ключевым фактором в процессе разработки. Фреймворк React Native признан перспективным и доминирующим.

А. Качмарчик и др. (Kaczmarczyk, A., Zając, P., & Zabierowski, W., 2022) [6] выполнили сравнение между нативными и гибридными мобильными приложениями для операционной системы Android, с акцентом на использование BLE (Bluetooth с низким энергопотреблением) и Wi-Fi (точность беспроводной связи). Было отмечено, что мобильные приложения становятся все более популярными в эпоху смартфонов и планшетов. Авторы провели сравнительный анализ, направленный на определение эффективности обработки данных в обеих технологиях.

Исследование P. Lachgar и соавторов (Lachgar, M., Hanine, M., Benouda, H., & Ommane, Y., 2022) [9] решает проблему разработки кроссплатформенных мобильных приложений в связи с растущей популярностью мобильных устройств. Авторы предлагают новую структуру для выбора оптимальной платформы для кроссплатформенной разработки, используя многокритериальные методы принятия решений, такие как аналитический иерархический процесс (AHP) и метод упорядочения предпочтений по сходству с идеальным решением (TOPSIS).

Постановка задачи. Целью исследования является проведение сравнительного анализа различных фреймворков для разработки мобильных приложений, а также сравнить их, учитывая ограничения и преимущества. Также были рассмотрены наиболее популярные фреймворки и инструменты для каждого подхода, чтобы предоставить объективную информацию, которая помогла бы разработчикам и организациям сделать осознанный выбор в области разработки мобильных приложений.

Методы исследования. В данном исследовании проводился сравнительный анализ фреймворков для разработки мобильных приложений с использованием нативных, гибридных и кроссплатформенных решений. Для достижения поставленной цели были использованы следующие методы: анализ, синтез и сравнение. Использование этих методов способствовало выяснению преимуществ и ограничений каждого из рассмотренных подходов в разработке мобильных приложений.

В рамках исследования для анализа были определены две основные мобильные платформы: Android и iOS, которые являются лидерами на современном рынке мобильных устройств. В ходе анализа было рассмотрено три основных подхода к разработке мобильных приложений: нативный, гибридный и кроссплатформенный. Для исследования нативных приложений использовались специализированные интегрированные среды разработки (IDE), такие как Android Studio для Android и Xcode для iOS.

Для изучения гибридных приложений использовались фреймворки, основанные на веб-технологиях, включая Apache Cordova, Ionic и React Native. Были рассмотрены кроссплатформенные фреймворки приложений, такие как Flutter, Xamarin и Appcelerator Titanium. Одним из основных критериев отбора фреймворков было их популярность и актуальность на рынке разработки мобильных приложений.

Исследование основано на анализе открытой научной литературы, которая была доступна в Интернете. Были использованы различные работы, исследования и публикации, в которых подробно описывались различные подходы и фреймворки для разработки мобильных приложений (Biørn-Hansen et al., 2020; Zohud & Zein, 2021; Lachgar et al., 2022) [2]. Анализ был сосредоточен на нескольких ключевых аспектах, включая производитель-

ность разработки, качество и быстродействие создаваемых приложений, стоимость разработки, поддержку различных мобильных платформ, расширяемость и другие важные факторы. Для каждого из рассмотренных подходов (нативных, гибридных и кроссплатформенных решений) были определены и проанализированы их преимущества и ограничения, что позволило провести объективный сравнительный анализ. Метод обобщения включал сбор информации о каждом из рассмотренных фреймворков, включая используемые языки программирования, доступность средств разработки, поддержку платформы, качество приложений, расширяемость и другие характеристики. Этот метод позволил получить полное представление о каждом подходе. Сравнение проводилось путем сопоставления полученных результатов для определения преимуществ и недостатков каждого подхода. При сравнительном анализе учитывались такие факторы, как производительность разработки, стоимость разработки, качество и производительность приложений, поддержка различных мобильных платформ, потребление памяти, расширяемость и другие параметры. На основе этого сравнительного анализа были выявлены сильные и слабые стороны каждого из рассмотренных подходов к разработке мобильных приложений. Выбор методов анализа, синтеза и сравнения для этого исследования оправданы тем, что они позволяют провести всесторонний и объективный анализ различных подходов к разработке мобильных приложений.

Метод анализа позволил глубже изучить каждый подход, выявив его особенности, технические параметры и другие характеристики. Обобщение собранной информации позволило получить полное представление о каждом подходе, что было полезно для дальнейшего сравнения. Сравнение выявило преимущества и недостатки различных подходов с учетом различных аспектов, такие, как производительность, стоимость разработки, приложение качество и производительность, поддержка различных мобильных платформ и другие факторы.

Обсуждение результатов. Нативные фреймворки для мобильных платформ – это специально разработанные инструменты, которые позволяют разработчикам создавать приложения, оптимизированные для конкретной платформы или операционной системы. Эти платформы предоставляют доступ к собственным возможностям и функциональным возможностям платформы, что приводит к разработке высокопроизводительных и эффективных программных продуктов. Ключевой особенностью этих платформ является их ориентация на разработку для конкретной платформы, такой как iOS или Android. Это означает, что разработчики имеют возможность в полной мере использовать все возможности платформы для создания приложений, которые работают оптимально и надежно. Как правило, для iOS используется язык программирования Swift и инструмент разработки Xcode, в то время как для Android это язык программирования Kotlin или Java и инструмент разработки Android Studio.

Xcode - это интегрированная среда разработки (IDE) предназначен для создания программного обеспечения для платформ iOS и macOS. Он специально разработан для языков программирования, рекомендованных Apple, включая C, C++, Objective-C, Swift, Java, AppleScript, Python и Ruby. Xcode предлагает расширенные возможности и инструменты, облегчающие процесс разработки, включая функции рефакторинга кода (Tkachuk & Bulakh, 2022) [16].

Современный Язык программирования Swift был разработан Apple для платформ iOS, macOS, watchOS и tvOS. Он призван обеспечить безопасность, производительность и выразительность кода, предлагая более удобную и эффективную альтернативу языку Objective-C (Fojtik, R., 2019) [4]. Swift интегрирован со средой разработки Xcode в операционной системе macOS. Такая интеграция позволяет разработчикам разрабатывать, тестировать и отлаживать свои программы Swift в единой среде. Xcode предоставляет ряд инструментов и функций, упрощающих процесс разработки, таких как конструктор визуального интерфейса, отладчик и симулятор для тестирования приложений

на различных устройствах (Ziyodullayevich, A.Q., Babakulovich, Z.R., Bakhodirovna, M.Z., Bekmurodovich, S.A., & Akif-ogli, M.R., 2019) [19]. Преимущество использования Xcode и Swift для разработки приложений для iOS заключается в возможности использования встроенных функций устройств Apple. Встроенная разработка позволяет создавать приложения, оптимизированные для конкретного аппаратного и программного обеспечения iOS, что приводит к повышению производительности и удобства использования. Однако это требует от разработчиков знания различных языков программирования и сред разработки (Biørn-Hansen, A., Rieger, C., Grønli, T.M., Majchrzak, T.A., & Ghinea, G., 2020) [2].

Основными языками программирования для разработки приложений для Android являются Java и Kotlin. Java – традиционный язык программирования для Android, в то время как Kotlin – это современная альтернатива, которая обеспечивает эффективность и удобство для разработчиков. Android Studio служит интегрированной средой разработки (IDE), специально разработана для создания приложений для Android. Она поддерживает оба основных языка программирования, Java и Kotlin, и предоставляет разработчикам необходимые инструменты для разработки и отладки приложений. Кроме того, для сценариев, требующих максимальной производительности, доступен Native Development Kit (NDK). Разработка программного обеспечения для Android Kit (SDK) предлагает инструменты и ресурсы для разработки приложений, тестирования и документации для Android. Он также включает в себя эмуляторы для тестирования приложений на различных устройствах и версиях Android (Sun, C., Ma, Y., Zeng, D., Tan, G., Ma, S., & Wu, Y., 2021) [14]. Все эти компоненты инфраструктуры разработки приложений для Android позволяют разработчикам создавать эффективные и надежные приложения для различных устройств, работающие на оптимальном уровне производительности и использующие мощные возможности платформы Android.

Разработка нативных мобильных приложений имеет несколько ключевых особенностей, отличающих его от других подходов к разработке. Одним из главных преимуществ является возможность использовать весь потенциал базовой операционной системы и аппаратного обеспечения устройства. Собственные приложения имеют прямой доступ к специфичным для устройства функциям и API (интерфейсам прикладного программирования), что позволяет разработчикам создавать оптимизированные и продуктивные приложения. Такой уровень контроля позволяет разрабатывать многофункциональные и привлекательные пользовательские интерфейсы с плавной анимацией, быстрым откликом и доступ к расширенным функциональным возможностям (Dittrich, F., Albrecht, U.V., Scherer, J., Becker, S.L., Landgraeber, S., Back, D.A., Fessmann, K., ... Klietz, M.L., 2023) [3]. Еще одной ключевой особенностью разработки нативных приложений является возможность следовать конкретным рекомендациям по проектированию компонентов пользовательского интерфейса. Собственные приложения могут обеспечить согласованное взаимодействие с пользователем, придерживаясь принципов проектирования и шаблонов целевой операционной системы (Thamutharam, Y.N., Mustafa, M.B.P., Musthafa, F.N., & Tajudeen, F.P., 2021) [15]. Это гарантирует, что приложение выглядит и ощущается как родное для платформы, что повышает удобство работы и удовлетворенность пользователей.

Разработка нативных приложений также обеспечивает лучшую интеграцию с экосистемой устройств. Разработчики могут легко получить доступ к таким возможностям устройств, как камера, GPS, акселерометр и push-уведомления (Raeesi, A., Khajouei, R., & Ahmadian, L., 2022) [12]. Это позволяет создавать приложения, которые могут в полной мере использовать потенциал устройства, предоставляя такие функции, как сервисы на основе определения местоположения, дополненная реальность и функциональность в режиме реального времени. Кроме того, по сравнению с другими подходами к разработке, нативные приложения обычно обеспечивают более высокую производительность. Поскольку они создаются с использованием нативных языков программирования и инструментов, с которыми они могут работать более эффективно. Однако разработка

нативных приложений также имеет ограничения и проблемы. Одним из основных недостатков является необходимость разработки отдельных кодовых баз для каждой целевой платформы (Masaad Alsaid, M.A.M., Ahmed, T.M., Sadeeq, J., Khan, F.Q., Mohammad, & Khattak, A.U., 2021) [11]. Это может увеличить время разработки и затраты. Кроме того, обновления и исправления ошибок могут потребовать отдельного развертывания для каждой платформы, что приведет к увеличению затрат на техническое обслуживание.

Платформы для разработки гибридных мобильных приложений. Гибридная разработка мобильных приложений позволяет разработчикам создавать мобильные приложения с использованием технологий веб-разработки, таких как HTML, CSS и JavaScript. Этот метод сочетает в себе преимущества других подходов и предлагает компромисс между нативными приложениями и веб-приложениями, позволяя создавать приложения, которые одновременно обеспечивают высокую функциональность для конкретной платформы и совместимость с различными платформами (Wu, C., Pérez-Álvarez, J.M., Mos, A., & Carroll, J.M.) [18].

Одним из самых популярных инструментов для гибридной разработки является Apache Cordova. Он инициализирует нативное приложение с помощью WebView, встроенного веб-браузера. Он служит связующим звеном между нативным кодом и веб-компонентами приложения, позволяя разработчикам писать бизнес-логику в JavaScript и создавать пользовательские интерфейсы с использованием HTML и CSS. Такой подход позволяет разрабатывать приложения, которые могут работать на различных платформах, включая Android и iOS, используя единую кодовую базу. Гибридные платформы разработки приложений предоставляют широкий спектр функций и возможностей для упрощения процесса разработки. Эти фреймворки часто включают компоненты и библиотеки, которые позволяют разработчикам создавать визуально привлекательные и быстрые пользовательские интерфейсы (Singh, M., & Shobha, G., 2021) [13]. Они также предоставляют доступ к функциям устройств и API-интерфейсам с помощью плагинов, позволяя разработчикам использовать встроенную функциональность в своих гибридных приложениях.

Помимо Apache Cordova, другие популярные гибридные фреймворки включают Ionic (который использует Angular) и React Native (основанный на React). Каждый из этих фреймворков обладает своими уникальными функциями и приложениями (табл.1), что позволяет разработчикам выбирать тот, который наилучшим образом соответствует их потребностям.

Таблица 1. Сравнительные характеристики популярных гибридных фреймворков
Table 1. Comparative characteristics of popular hybrid frameworks

Фреймворк **Framework**	**Описание** **Description**	**Языки программирования** **Programming Languages**	**Основные характеристики** **Main Features**
Apache Cordova	Использует веб-технологии для разработки мобильных приложений.	HTML, CSS, JavaScript	Поддерживает множество плагинов для расширения функциональности устройства
Ionic	Гибридный фреймворк, основанный на Angular. Разработан специально для создания красивых и функциональных мобильных приложений	HTML, CSS, JavaScript (с Angular)	Готовые компоненты и инструменты для пользовательского интерфейса
React Native	Позволяет использовать React для создания мобильных приложений с интерфейсом и функциональностью нативных приложений	JavaScript (с React)	Возможность повторного использования кода на разных платформах и высокая производительность.

Одним из главных преимуществ гибридной разработки является возможность использовать существующие навыки и ресурсы веб-разработчиков. Разработчики, владеющие веб-технологиями, могут легко перейти к гибридной разработке, поскольку они могут использовать свои знания HTML, CSS и JavaScript. Это может привести к сокращению сроков разработки и снижению затрат по сравнению с разработкой отдельных нативных приложений для каждой платформы. Однако гибридная разработка также имеет свои недостатки. Поскольку гибридные приложения используют WebView для отображения пользовательского интерфейса, они могут не обеспечивать такого же уровня производительности, как нативные приложения (Hu, J., Wei, L., Liu, Y., & Cheung, S.C., 2023) [5]. Кроме того, для доступа к определенным функциям устройства и API-интерфейсам может потребоваться использование подключаемых модулей, что создает дополнительную сложность и потенциальные проблемы с совместимостью. В целом, разработка гибридных мобильных приложений представляет собой компромисс между нативными и веб-приложениями. Это позволяет разработчикам создавать приложения для разных платформ с использованием веб-технологий, предоставляя доступ к функциональным возможностям устройств. Apache Cordova - популярный инструмент для гибридных приложений. разработка, позволяющая использовать HTML, CSS и JavaScript для создания приложений, которые могут работать на различных платформах. Хотя гибридные приложения могут отличаться производительностью от нативных приложений, они обладают преимуществами с точки зрения эффективности разработки и экономичности.

Разработка кроссплатформенных мобильных приложений – это современный подход в области программного обеспечения для мобильных устройств. Он предполагает создание приложений, которые могут работать на разных платформах, таких как Android и iOS, с использованием единой кодовой базы и других общих ресурсов. Кроссплатформенная разработка приобрела популярность благодаря возможности эффективного использования общего кода для создания приложений на разных платформах. Это позволяет разработчикам экономить время и ресурсы, избегая необходимости поддерживать отдельный код для каждой платформы. Основным преимуществом кроссплатформенной разработки является снижение затрат на разработку и обслуживание приложений, поскольку общий код может использоваться на всех платформах. Кроме того, разработчики могут использовать единый язык программирования для создания приложений.

Кроссплатформенные платформы, такие как Flutter, Xamarin и другие (табл. 2), предоставляют инструменты для разработки мобильных приложений, внешний вид и поведение которых аналогичны нативным приложениям на каждой платформе (Martínez, M., 2019) [10]. Это обеспечивает высокое качество и согласованную функциональность в различных операционных системах.

Таблица 2. Обзор популярных кроссплатформенных платформ
Table 2. Overview of popular cross-platform platforms

Фреймворк Framework	**Языки программирования Programming Languages**	**Описание Description**
Flutter	Dart	Фреймворк от Google для создания красивых мобильных приложений с единой кодовой базой и использованием собственного движка рендеринга.
Xamarin	C#	Фреймворк от Microsoft, позволяющий разработчикам использовать язык программирования C# для создания мобильных приложений для Android и iOS.
Appcelerator Titanium	JavaScript	Фреймворк, использующий JavaScript для разработки кроссплатформенных мобильных приложений и предоставляющий доступ к встроенным функциям.

Кроссплатформенные фреймворки для разработки мобильных приложений, предлагая такие преимущества, как унифицированная кодовая база и снижение затрат на разработку, также обладают рядом недостатков.

Во-первых, одним из основных недостатков является ограниченный доступ к встроенным функциям и API-интерфейсам платформы. Кроссплатформенные платформы пытаются абстрагировать функциональность устройства, но иногда они могут не полностью воспроизводить все возможности, доступные на конкретной платформе. Вторым недостатком является производительность и скорость.

Разработанные мобильные приложения с использованием кроссплатформенных платформ могут замедлить работу и потребовать больше ресурсов по сравнению с собственными приложениями. Кроме того, задержки с обновлениями могут привести к трудностям при использовании новых функций. Наконец, кроссплатформенные платформы могут быть не так хорошо адаптированы к специфическому внешнему виду и поведению определенных платформ, что может потребовать дополнительной настройки и адаптации. Таким образом, кроссплатформенные фреймворки, хотя и позволяют экономить время и ресурсы при разработке мобильных приложений, имеют ограничения и недостатки, которые необходимо учитывать при выборе подхода к разработке.

В современных условиях разработки мобильных приложений выбор оптимального подхода имеет решающее значение. Разработчики сталкиваются с различными возможностями и ограничениями при выборе между нативными, гибридными и кроссплатформенными решениями.

В табл. 3 представлен сравнительный анализ этих подходов к разработке. Нативные приложения отличаются высокой производительностью и скоростью благодаря использованию нативных компонентов и языков программирования для каждой платформы в отдельности. Для разработки нативных приложений на платформе Android используются такие языки программирования, как Java или Kotlin, в то время как на платформе iOS используются Swift или Objective-C. Такой подход обеспечивает полный доступ ко всем возможностям устройства и встроенным API, позволяющие создавать приложения с отличной производительностью и скоростью. Однако для этого требуется больше ресурсов и времени на разработку. Необходимость в отдельном коде для каждой платформы приводит к увеличению затрат и продолжительности проекта.

С другой стороны, гибридные приложения позволяют сократить затраты и время разработки за счет использования единой кодовой базы для обеих платформ. Они часто полагаются на веб-технологии, такие как HTML, CSS и JavaScript, что упрощает разработку для опытных веб-разработчиков. Одной из популярных платформ для гибридной разработки является Apache Cordova, которая инициализирует нативное приложение с помощью WebView, встроенного веб-браузера. Гибридные приложения могут демонстрировать более низкую производительность из-за использования WebView и ограниченный доступ к возможностям устройства. Плагины могут использоваться для доступа к определенным функциям устройства, таким как камера или геолокация.

Кроссплатформенные приложения сочетают в себе преимущества обоих предыдущих подходов, позволяя использовать единую кодовую базу для обеих платформ. При правильном выборе кроссплатформенного фреймворка можно добиться удовлетворительной производительности. Одним из таких фреймворков является Flutter, разработанный Google, который использует язык программирования Dart и собственный движок для рендеринга пользовательского интерфейса. Еще одним популярным вариантом является фреймворк Xamarin от Microsoft, который позволяет использовать язык программирования C#. Кроссплатформенные приложения могут также использовать специализированные платформы для улучшения дизайна и внешнего вида приложений. Однако они могут выглядеть менее «нативно» и иметь ограниченный доступ к функциям, зависящим от платформы.

Таблица 3. Сравнительный анализ нативных, гибридных и кроссплатформенных решений для разработки мобильных приложений
Table 3. Comparative analysis of native, hybrid and cross-platform solutions for mobile application development

Параметр Parameter	**Нативные приложения Native applications**	**Гибридные приложения Hybrid applications**	**Кроссплатформенные приложения Cross-platform applications**
Производительность и скорость	Высокая производительность, оптимизированная скорость работы с использованием собственных компонентов и языков программирования Java/Kotlin для Android, Swift/Objective-C для iOS.	Умеренная производительность и скорость благодаря использованию WebView для отображения пользовательского интерфейса и некоторой абстракции.	Производительность умеренная, но может быть улучшена с помощью специализированных кроссплатформенных платформ (например, Flutter).
Стоимость и время разработки	Высокие затраты и более длительное время разработки из-за необходимости создавать отдельный код для каждой платформы.	Снижение затрат и ускорение разработки благодаря использованию единой кодовой базы для обеих платформ. Некоторые приложения могут потребовать корректировки для конкретной платформы.	Более низкие затраты и меньше времени, но разработка может занять немного больше времени по сравнению с гибридными решениями из-за выбора кроссплатформенной платформы и обучения команды.
Доступ к возможностям устройства	Полный доступ ко всем возможностям устройства и встроенным API.	Ограниченный доступ, требует использования плагинов для доступа к определенным функциям устройства.	Средний доступ; кроссплатформенные платформы предоставляют API для упрощенного доступа к возможностям устройств, но не всегда могут обладать полным набором функций.
Внешний вид	Оригинальный внешний вид на каждой платформе обеспечивает высокое качество и безупречный дизайн.	Менее привычный внешний вид; интерфейс может быть менее привлекательным и в меньшей степени соответствовать стандартам дизайна каждой платформы.	Как правило, менее привычный внешний вид, но возможность использовать специализированные фреймворки для улучшения дизайна.
Поддержка платформы	Зависит от платформы (Android и iOS).	Обе платформы (Android и iOS) основаны на единой кодовой базе.	Обе платформы (Android и iOS) основаны на единой кодовой базе
Обновления и поддержка	Отдельно для каждой платформы, отдельно для обновлений и обслуживания.	Обновления и техническое обслуживание из одной кодовой базы, но может возникнуть необходимость в обновлениях для каждой ОС (операционной системы) отдельно для поддержки новых функций платформы.	Обновления и обслуживание из единой кодовой базы, возможность дополнительной работы по поддержке новых функций.

Вывод. В ходе исследования были рассмотрены различные подходы к разработке мобильных приложений, такие как нативные, гибридные и кроссплатформенные решения. Нативные приложения, разработанные для конкретной платформы, обеспечивают высочайшую производительность и доступ к возможностям устройства, но требуют значитель-

ных усилий и ресурсов. Гибридные приложения сокращают затраты и время за счет использования общей кодовой базы, но могут иметь ограниченный доступ к функциям устройства. Кроссплатформенные приложения, сочетающие преимущества обоих подходов, могут быть эффективными благодаря правильному выбору фреймворка.

В ходе исследования были проанализированы различные фреймворки для разработки мобильных приложений, включая Flutter, Xamarin, Apache Cordova, React Native и Ionic. У каждого из них есть свои особенности и преимущества. Flutter, разработанный Google, позволяет создавать приложения с высококачественным интерфейсом и высокой производительностью с помощью единой кодовой базы. Xamarin от Microsoft использует язык программирования C# и интегрируется с экосистемой Microsoft. Apache Cordova основан на веб-технологиях и инициализирует нативное приложение через WebView. React Native от Facebook позволяет создание приложений с нативным дизайном и быстрым кодированием с использованием React и JavaScript. Ionic использует HTML, CSS и JavaScript для гибридной разработки с возможностью использования Angular. Результаты исследования показывают, что выбор фреймворка должен основываться на конкретных требованиях к проекту с учетом производительности, бюджета и ресурсов. Каждый фреймворк имеет свои преимущества и ограничения, и важно учитывать их в контексте конкретного проекта. Рекомендации включают выбор подхода к разработке мобильного приложения на основе конкретных требований к проекту, финансовых ограничений и ресурсов.

Библиографический список:

1. Альрабайя, Х.А., и Медина-Медина, Н. (2021). Agile beeswax: процесс разработки мобильных приложений и эмпирическое исследование в реальной среде. Устойчивое развитие, 13(4), артикул 1909. DOI: 10.3390/su13041909.
2. Бьерн-Хансен А., Ригер К., Гренли Т.М., Майчшак Т.А. и Гинея Г. (2020). Эмпирическое исследование повышения производительности кроссплатформенных платформ мобильной разработки. Эмпирическая разработка программного обеспечения, 25, 2997-3040. DOI: 10.1007/s10664-020-09827-6.
3. Дитрих Ф., Альбрехт У.В., Шерер Дж., Беккер С.Л., Ландграбер С., Бэк Д.А., Фессманн К., ... Клиц, М.Л. (2023). Разработка открытых серверных структур для медицинских работников с целью улучшения участия в разработке приложений: пилотное исследование удобства использования медицинского приложения. JMIR Formative Research, 7, номер статьи e42224. DOI:10.2196/42224.
4. Фойтик Р. (2019). Swift - новый язык программирования для разработки и образования. В работе Т. Антиповой и А. Роша (ред.), Digital Science 2019 (том 1114; стр. 284-295). Издательство: Springer. DOI: 10.1007/978-3-030-37737-3_26.
5. Ху, Дж., Вэй, Л., Лю, Ю. и Чунг, С.К. (2023). wTest: Webview-ориентированное тестирование приложений для Android. В ISSTA 2023: Материалы 32-го международного симпозиума ACM SIGSOFT по тестированию и анализу программного обеспечения (стр. 992-1004). Нью-Йорк: Ассоциация вычислительной техники. DOI: 10.1145/3597926.3598112.
6. Качмарчик А., Зайенц П. и Забьеровский В. (2022). Сравнение производительности нативных и гибридных мобильных приложений Android, основанных на сенсорных данных, основанных на коммуникационной архитектуре Bluetooth с низким энергопотреблением (BLE) и Wi-Fi. Energies, 15(13), артикул 4574. DOI: 10.3390/en15134574.
7. Каратанов О., Йена М., Бова Ю., Устименко О. (2021). Сравнение популярных тестовых фреймворков JUnit и TestNG. Молодой ученый, 5(93), 164-170. DOI: 10.32839/2304-5809/2021-5-93-31.
8. Козуб, Н., & Козуб, Ю. (2022). Декларативный метод создания мультиплатформенных приложений. Вестник Восточноукраинского национального университета имени Владимира Даля, 5(275), 10-15. DOI: 10.33216/1998-7927- 2022-275-5-10-15.
9. Лачгар, М., Ханин, М., Бенуда, Х. и Омман, Ю. (2022). Система принятия решений для кроссплатформенных платформ мобильной разработки, использующая интегрированную методологию принятия многокритериальных решений. Международный журнал мобильных вычислений и мультимедийных коммуникаций, 13 (1), 1-22. DOI: 10.4018/ijmcmc.297928.
10. Мартинес, М. (2019). Два набора вопросов и ответов для изучения разработки кроссплатформенных мобильных приложений с использованием платформы Xamarin framework. В 2019 году состоится 6-я международная конференция IEEE/ACM по разработке мобильного программного обеспечения

и систем (MOBILESoft) (стр. 162-173). Пискатауэй: Институт инженеров электротехники и электроники. DOI: 10.1109/mobilesoft.2019.00032.

11. Масаад Альсаид, М.А.М., Ахмед, Т.М., Садик, Дж., Хан, Ф.К., Мохаммад и Хаттак, А.У. (2021). Сравнительный анализ подходов к разработке мобильных приложений: Подходы к разработке мобильных приложений. Труды Пакистанской академии наук: А. Физические и вычислительные науки, 58 (1), 35-45. DOI: 10.53560/PPASA(58-1)717.
12. Раиси, А., Хаджуи, Р. и Ахмадиан, Л. (2022). Оценка и ранжирование мобильных приложений для борьбы с ВИЧ/СПИДом с использованием метода оценки приложений на основе функций и шкалы оценки мобильных приложений. BMC Medical Informatics and Decision Making, 22, статья № 281. DOI: 10.1186/s12911-022-02029-8.
13. Сингх, М., и Шобха, Г. (2021). Сравнительный анализ платформ разработки гибридных мобильных приложений. Международный журнал мягких вычислений и инженерии, 10 (6), 21-26. DOI: 10.35940/ijsce.f3518.0710621.
14. Sun, C., Ma, Y., Zeng, D., Tan, G., Ma, S., & Wu, Y. (2021). μDep: Генерация зависимостей на основе мутаций для точного анализа дефектов в машинном коде Android. В IEEE transactions on dependable and secure computing, 20 (2), 1461-1475. Пискатауэй: Институт инженеров электротехники и электроники. DOI: 10.1109/TDSC.2022.3155693.
15. Тамутарам Ю.Н., Мустафа М.Б.П., Мустафа Ф.Н. и Таджудин Ф.П. (2021). Функции юзабилити для улучшения восприятия мобильных приложений пожилыми гражданами Малайзии. Научный журнал ASM, 16. DOI: 10.32802/ asmscj.2021.686.
16. Ткачук А., Булах Б. (2022). Исследование возможностей действий по рефакторингу по умолчанию в языке swift. Технологический аудит и резервы производства, 5(2(67)), 6-10. DOI: 10.15587/2706-5448.2022.266061.
17. Uplenchwar, S.R., Denge, США, Bajoriya, A.S. и Bachwani, S.A. (2022). Обзор подробной информации о кросс-платформе flutter. Международный журнал исследований в области прикладных наук и инженерных технологий, 10 (1), 1016-1022. DOI: 10.22214/ijraset.2022.39977.
18. Ву, К., Перес-Альварес, Дж.М., Мос, А., и Кэрролл, Дж.М. (2022). Разработка приложений без кода: оценка подхода к облачным функциям с учетом специфики предметной области. В материалах 56-й ежегодной Гавайской международной конференции по системным наукам, HICSS 2023 (стр. 6904-6913). Вашингтон: Компьютерное общество IEEE. DOI: 10.48550/arxiv.2210.01647.
19. Зиедуллаевич А.К., Бабакулович З.Р., Баходировна М.З., Бекмуродович С.А. и Акиф-оглы М.Р. (2019). Эффективное и удобное приложение для определения функций и анализа надежности устройства. Международный журнал инновационных технологий и инженерных исследований, 9 (2), 1804-1809. DOI: 10.35940/ijitee. b7323.129219.
20. Зохуд, Т., и Зейн, С. (2021). Разработка кроссплатформенных мобильных приложений в промышленности: несколько примеров из практики.

References:

1. Alrabaiah, H.A., & Medina-Medina, N. (2021). Agile beeswax: Mobile app development process and empirical study in real environment. Sustainability, 13(4), article number 1909. DOI: 10.3390/su13041909.
2. Biørn-Hansen, A., Rieger, C., Grønli, T.M., Majchrzak, T.A., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. Empirical Software Engineering, 25, 2997-3040. DOI: 10.1007/s10664-020-09827-6.
3. Dittrich, F., Albrecht, U.V., Scherer, J., Becker, S.L., Landgraeber, S., Back, D.A., Fessmann, K., ... Klietz, M.L. (2023). Development of open backend structures for health care professionals to improve participation in app developments: Pilot usability study of a medical app. JMIR Formative Research, 7, article number e42224. DOI: 10.2196/42224.
4. Fojtik, R. (2019). Swift a new programming language for development and education. In T. Antipova & Á. Rocha (Eds.), Digital Science 2019; 1114: 284-295). Cham: Springer. DOI: 10.1007/978-3-030-37737-3_26.
5. Hu, J., Wei, L., Liu, Y., & Cheung, S.C. (2023). ωTest: Webview-oriented testing for android applications. In ISSTA 2023: Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis 2023:992-1004). New York: Association for Computing Machinery. DOI: 10.1145/3597926.3598112.
6. Kaczmarczyk, A., Zając, P., & Zabierowski, W. (2022). Performance comparison of native and hybrid android mobile applications based on sensor data-driven applications based on Bluetooth low energy (BLE) and Wi-Fi communication architecture. Energies, 15(13), article number 4574. DOI: 10.3390/en15134574.
7. Karatanov, O., Yena, M., Bova, Y., & Ustymenko, O. Comparison of popular test frameworks JUnit and TestNG. Young Scientist, 2021:5(93), 164-170. DOI: 10.32839/2304-5809/2021-5-93-31. (In Russ)

8. Kozub, H., & Kozub, Yu. (2022). Declarative method for creating multiplatform applications. Bulletin of the Eastern Ukrainian National University named after Volodymyr Dahl, 5(275), 10-15. DOI: 10.33216/1998-7927- 2022-275-5-10-15.
9. Lachgar, M., Hanine, M., Benouda, H., & Ommane, Y. (2022). Decision framework for cross-platform mobile development frameworks using an integrated multi-criteria decision-making methodology. International Journal of Mobile Computing and Multimedia Communications, 13(1), 1-22. DOI: 10.4018/ijmcmc.297928.
10. Martínez, M. (2019). Two datasets of questions and answers for studying the development of cross-platform mobile applications using Xamarin framework. In 2019 IEEE/ACM 6th international conference on mobile software engineering and systems (MOBILESoft) (pp. 162-173). Piscataway: Institute of Electrical and Electronics Engineers. DOI: 10.1109/mobilesoft.2019.00032.
11. Masaad Alsaid, M.A.M., Ahmed, T.M., Sadeeq, J., Khan, F.Q., Mohammad, & Khattak, A.U. (2021). A comparative analysis of mobile application development approaches: Mobile application development approaches. Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences, 58(1), 35-45. DOI: 10.53560/PPASA(58-1)717.
12. Raeesi, A., Khajouei, R., & Ahmadian, L. (2022). Evaluating and rating HIV/AIDS mobile apps using the feature-based application rating method and mobile app rating scale. BMC Medical Informatics and Decision Making, 22, article number 281. DOI: 10.1186/s12911-022-02029-8.
13. Singh, M., & Shobha, G. (2021). Comparative analysis of hybrid mobile app development frameworks. International Journal of Soft Computing and Engineering, 10(6), 21-26. DOI: 10.35940/ijsce.f3518.0710621.
14. Sun, C., Ma, Y., Zeng, D., Tan, G., Ma, S., & Wu, Y. (2021). μDep: Mutation-based dependency generation for precise taint analysis on android native code. In IEEE transactions on dependable and secure computing, 2021:20(2):1461-1475. Piscataway: Institute of Electrical and Electronics Engineers. DOI: 10.1109/TDSC.2022.3155693.
15. Thamutharam, Y.N., Mustafa, M.B.P., Musthafa, F.N., & Tajudeen, F.P. (2021). Usability features to improve mobile apps acceptance among the senior citizens in Malaysia. ASM Science Journal, 16. DOI: 10.32802/ asmscj.2021.686.
16. Tkachuk, A., & Bulakh, B. (2022). Research of possibilities of default refactoring actions in swift language. Technology Audit and Production Reserves, 2022;5(2(67)):6-10. DOI: 10.15587/2706-5448.2022.266061.
17. Uplenchwar, S.R., Denge, U.S., Bajoriya, A.S., & Bachwani, S.A. (2022). Review on detail information about flutter cross platform. International Journal for Research in Applied Science and Engineering Technology, 2022;10(1):1016-1022. DOI: 10.22214/ijraset.2022.39977.
18. Wu, C., Pérez-Álvarez, J.M., Mos, A., & Carroll, J.M. (2022). Codeless app development: Evaluating a cloud-native domain-specific functions approach. In Proceedings of the 56th annual Hawaii international conference on system sciences, HICSS 2022:6904-6913). Washington: IEEE Computer Society. DOI: 10.48550/arxiv.2210.01647.
19. Ziyodullayevich, A.Q., Babakulovich, Z.R., Bakhodirovna, M.Z., Bekmurodovich, S.A., & Akif-ogli, M.R. (2019). Efficient and convenient application to determine the functions and analysis of the reliability of the device. *International Journal of Innovative Technology and Exploring Engineering*, 2019;9(2):1804-1809. DOI: 10.35940/ijitee. b7323.129219.
20. Zohud, T., & Zein, S. (2021). Cross-platform mobile app development in industry: A multiple case-study.

Сведения об авторе:

Темирова Аза Бароновна, старший преподаватель кафедры «Информационные технологии», aza0109@mail.ru; ORCID - 0000-0003-1090-5180

Information about author:

Aza B. Temirova, Senior lecturer, Department of Information Technology, aza0109@mail.ru; ORCID - 0000-0003-1090-5180

Конфликт интересов/Conflict of interest.

Автор заявляет об отсутствии конфликта интересов/The author declare no conflict of interest.

Поступила в редакцию/Received 02.10.2024.

Одобрена после рецензирования/Reviced 30.11.2024.

Принята в печать/Accepted for publication 19.01.2025.