Flutter vs. React Native: Что лучше в 2023 году?

Описание

Что лучше выбрать для быстрой и экономичной разработки кроссплатформенного мобильного приложения в 2023 году? Мы сравнили две технологии – Flutter и React Native – чтобы вы могли определить, какой кроссплатформенный фреймворк лучше всего подойдет для вашего приложения.

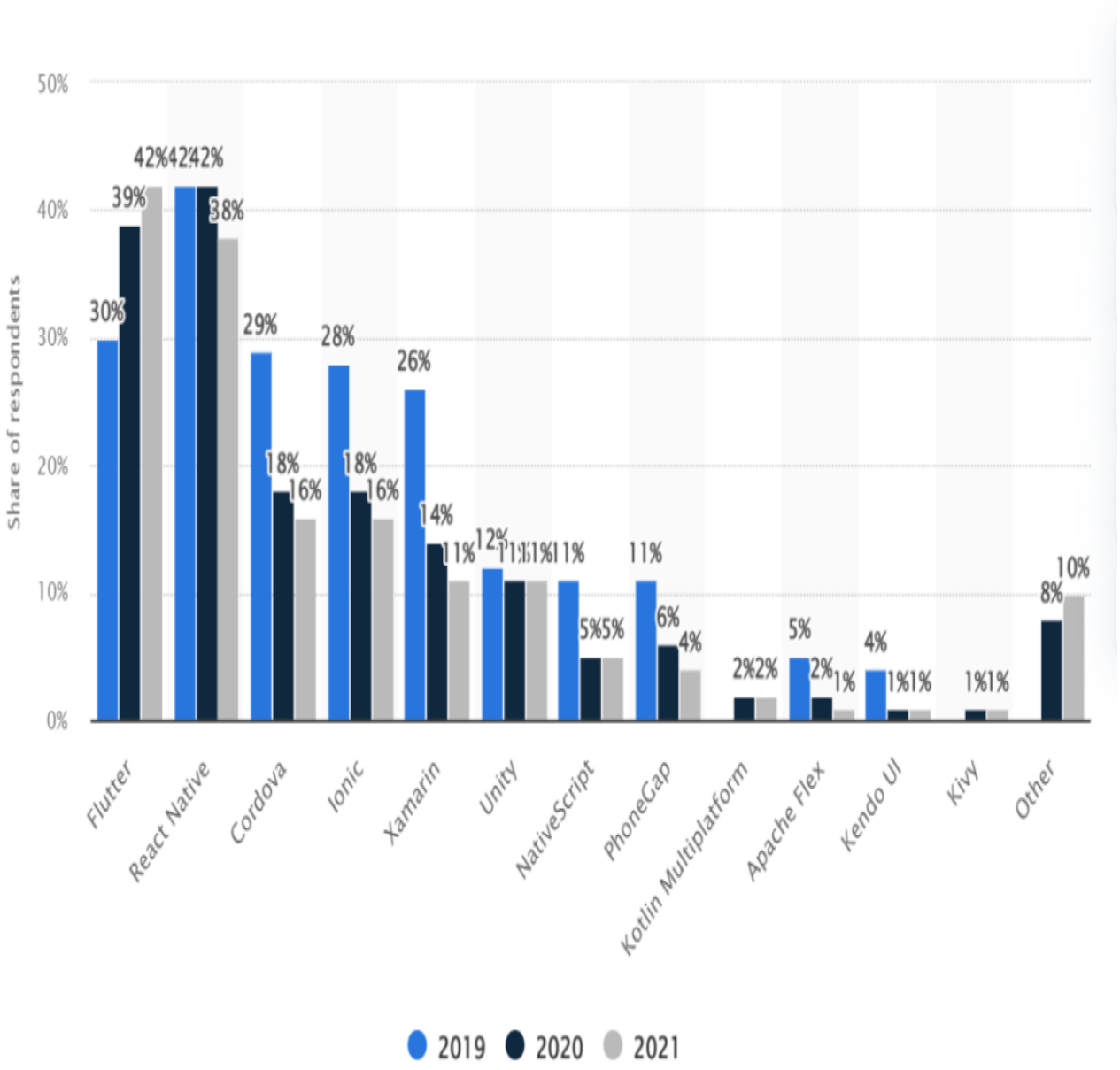
Обзор

Бизнес мобильных приложений неуклонно растет. Сегодня почти у каждого есть смартфон, поэтому количество возможных клиентов практически безгранично. Именно поэтому каждая компания стремится создать мобильное приложение, чтобы оставаться конкурентоспособной в своей нише.

Существует несколько способов разработки и создания приложений. Можно обратиться к нативным методам, то есть создать отдельное приложение для каждой платформы, используя разные языки (Java и Kotlin для Android; Swift и Objective-C для iOS). Но существует и практика кроссплатформенной разработки, когда инженеры-программисты создают единый код, работающий на всех платформах для мобильных устройств. Для этого существуют специальные фреймворки, самыми популярными из которых в настоящее время являются Flutter и React Native.

Flutter набирает популярность среди разработчиков в течение последних нескольких лет и занимает первое место в рейтинге Statista в 2021 году. React

Native дышит ему в спину, занимая второе место.



В этой статье мы расскажем об особенностях Flutter и React Native, а также об их плюсах и минусах.

Что такое Flutter?

Flutter – это достаточно молодой (появился в 2017 году) SDK от Google. Это бесплатный набор инструментов с открытым исходным кодом для разработки кроссплатформенных мобильных приложений, а также веб-, десктопных и встраиваемых решений из единой кодовой базы. Используя Flutter, вы можете

разрабатывать как MVP, так и полнофункциональные продукты.

Сам Flutter написан на языках C/C++ и Dart. Но для разработки приложений он использует Dart.

Dart на сегодняшний день является самым быстрым языком программирования для создания приложений для iOS и Android. Он помогает создавать хорошо функционирующие компоненты пользовательского интерфейса, имеет отличную поддержку IDE с мощными функциями автозавершения, позволяет создавать живые прототипы без потери состояния приложения и требует объектно-ориентированного программирования. Он позволяет создавать гибкие и выразительные пользовательские интерфейсы, которые работают в естественном режиме.

Flutter состоит из двух частей:

- SDK, который представляет собой набор инструментов для разработки, отладки, профилирования и многого другого.
- Сам фреймворк, который представляет собой библиотеку пользовательского интерфейса на основе виджетов. Он представлен коллекцией кнопок, текста, слайдеров и т.д., которые можно настраивать и создавать свой собственный уникальный дизайн.

Приложения на основе Flutter создаются из единой кодовой базы, а язык программирования Dart компилируется непосредственно в нативный код, что повышает производительность этих приложений. Они используют графический процессор (GPU) для рендеринга и могут обращаться к API и сервисам платформы (камера, локальное хранилище и многое другое).

Современное мобильное приложение должно иметь красивый дизайн, плавную анимацию и отличную производительность. Чтобы достичь этого, разработчики должны создавать функции быстрее, чем когда-либо, не жертвуя качеством или производительностью. Это и послужило причиной изобретения Flutter компанией Google. Когда Flutter был впервые продемонстрирован, разработчики были очень впечатлены тем, насколько хорошо он работает на мобильных устройствах по сравнению с другими технологиями, разработанными для кроссплатформенных мобильных приложений.

Плюсы и минусы Flutter

Почему разработчики выбирают Flutter? Flutter имеет множество преимуществ.

- **Быстрая разработка.** С помощью Flutter можно быстро разработать прототип или даже готовое приложение. Flutter для веб хорошо подходит для разработки прогрессивных веб-приложений (PWA), одностраничных приложений (SPA) и создания веб-версий существующих мобильных приложений.
- **Горячая перезагрузка.** Эта функция обеспечивает быструю компиляцию и максимальную производительность. Вы можете изменить какой-то элемент или виджет в проекте, и он мгновенно изменится в UI приложения, не влияя на состояние приложения.
- **Быстрая производительность.** Приложения на базе Flutter работают очень быстро, до 60 кадров в секунду и даже больше.
- **Богатый набор настраиваемых виджетов.** Flutter предлагает богатый выбор виджетов на основе Material Design (стиль Google) или фреймворка Cupertino (стиль Apple), благодаря которым приложения выглядят великолепно.
- **Интеграции.** Flutter интегрируется с популярными инструментами разработки, такими как Visual Studio Code, Android Studio и Xcode, что означает, что вы можете работать с вашим любимым редактором или IDE. Кроме того, приложения на Flutter могут быть легко интегрированы с сервисами Google, поскольку Google является создателем фреймворка.
- **Гибкий и выразительный пользовательский интерфейс.** Flutter предоставляет виджеты, рендеринг, анимацию и жесты, давая вам полный контроль над каждым пикселем на экране. Это означает, что вы можете создать оригинальный пользовательский дизайн для своего приложения.
- **Нативные приложения.** Приложения, написанные на Flutter, будут иметь универсальный пользовательский интерфейс как на устройствах Android, так и на iOS. Таким образом, вы можете разработать высокоадаптивный интерфейс для двух платформ: в Google Play Store они будут выглядеть как Material design, в App Store – как дизайн Купертино.
- **Открытый исходный код.** Поскольку Flutter предоставляет открытый исходный код, количество готовых решений для этого фреймворка постоянно растет. Вы можете использовать готовые решения в своих проектах или создавать что-то новое, тем самым внося свой вклад в развитие и

совершенствование проекта Flutter.

- **Легкость в освоении.** Изучение Flutter с нуля может занять не более 3-4 месяцев. Платформа основана на языке программирования Dart, синтаксис которого схож с широко используемыми C#, Java и JavaScript. Это означает, что изучение Flutter – простая задача для опытных программистов и даже для новичков.
- **Отличная документация.** И Flutter, и Dart снабжены подробной документацией, что упрощает их изучение и практическое использование.

Flutter отлично подходит для новичков: если вы новичок в мобильной разработке, выбирайте Flutter, поскольку он обеспечивает быстрый, интересный и современный способ создания нативных мобильных приложений. Если вы опытный разработчик, то у вас есть еще один увлекательный инструмент, который стоит попробовать.

Что касается недостатков, то можно назвать несколько.

- **Молодой проект.** Flutter – молодая технология, поэтому возможны баги, проблемы с обновлениями и все прочие “прелести” молодого инструмента.
- **Dart не является популярным языком.** Чтобы работать с Flutter, разработчикам придется потратить время на изучение нового языка и подходов к разработке. Поэтому в настоящее время число разработчиков Dart невелико, и поиск квалифицированных специалистов может оказаться непростой задачей.
- **Необходим мощный компьютер.** Dart создает свою собственную виртуальную машину, которая требует вычислительных ресурсов. Эмулятор и IDE также потребляют память и процессор. И если у вас слабый процессор и менее 8 ГБ оперативной памяти, работать с Flutter будет очень сложно.

Что такое React Native?

React Native – это мощный, бесплатный фреймворк с открытым исходным кодом. Это один из самых популярных JavaScript-фреймворков в мире на сегодняшний день. Как и Flutter, он поддерживает концепцию единой кодовой базы для нескольких операционных систем. Это позволяет разрабатывать кроссплатформенные приложения для iOS, Android, Windows и macOS.

Поскольку React Native использует язык JavaScript и библиотечную архитектуру React, он упрощает создание веб-версий приложений и обеспечивает достаточно

быстрое время разработки и исправления ошибок.

Следует отметить, что React Native не является обновленной версией библиотеки React, хотя и использует ее в качестве основы.

React (она же ReactJS) – это библиотека, используемая для создания интерфейса веб-сайта. Она была разработана командой инженеров Facebook в 2013 году. В то время как React Native, основанный на React, – это полнофункциональный фреймворк, запущенный в 2015 году (также командой инженеров Facebook), цель которого – дать разработчикам возможность использовать набор компонентов пользовательского интерфейса для быстрой компиляции и запуска приложений для iOS и Android.

Плюсы и минусы React Native

Каковы преимущества разработки React Native, и почему вы должны выбрать именно этот инструмент для создания своего мобильного приложения? Давайте посмотрим.

- **Экономическая эффективность.** React Native предлагает недорогой способ создания кроссплатформенных приложений за счет повторного использования кода. Приложения могут эффективно работать на нескольких платформах, что очень ценится владельцами продуктов. Вместо того чтобы создавать два разных приложения для Android и iOS, разработчики могут использовать до 90% одного и того же кода для обеих платформ. Это означает снижение затрат на разработку, а также делает React Native более дешевым в обслуживании.
- **Модульная конструкция.** В React Native используется технология модульного программирования, то есть функции реализуются в виде отдельных блоков, называемых модулями. Такой подход обеспечивает гибкую среду разработки приложений, а также улучшает сотрудничество между разработчиками. Он упрощает создание и интеграцию обновлений приложений.
- **Быстрое время выхода на рынок.** Способность React Native быстрее разрабатывать приложения – одна из самых привлекательных особенностей платформы. Разработчики используют множество готовых компонентов для создания функций приложения быстрее, чем когда-либо. React Native проще в кодировании, чем другие платформы разработки. Поэтому для создания и запуска приложений React Native требуется меньше усилий.
- **Горячая перезагрузка.** Эта функция является одной из самых полезных и

привлекательных для разработчиков. Она позволяет им вносить изменения в код в режиме реального времени. Таким образом, вы можете обновить работающее приложение. Для этого достаточно отредактировать исходный код, и обновление будет запущено в режиме реального времени после сохранения файла. Поэтому разработчики могут выпускать обновления без вынужденного простоя.

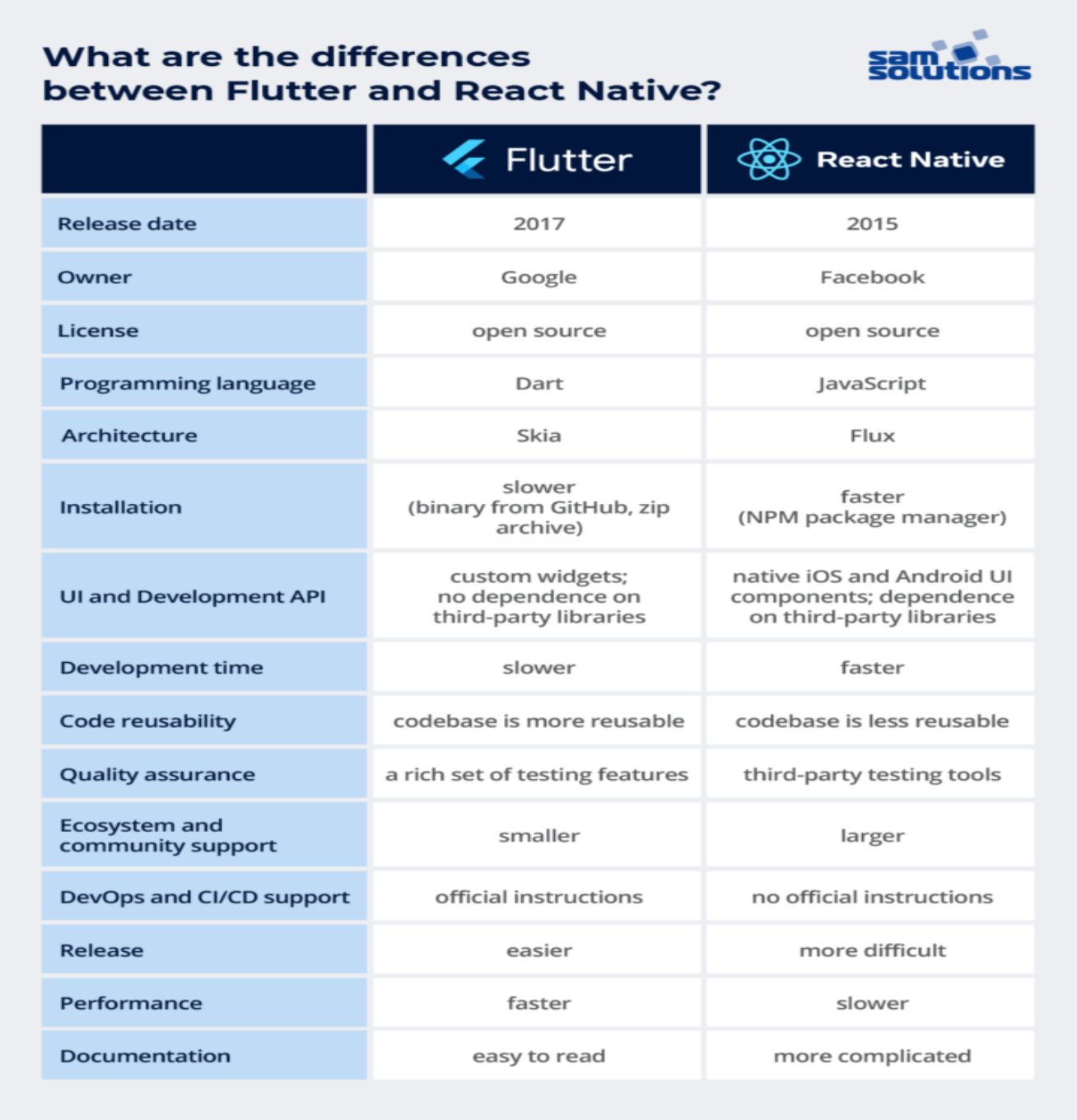
- **Отличная производительность.** Разработка нативных приложений обеспечивает более высокую производительность, но приложения, созданные с помощью React Native, также демонстрируют впечатляющую производительность, сравнимую с нативными приложениями. Это возможно благодаря встроенным элементам управления, которые используют нативные компоненты ОС для бесшовного кодирования нативных API.
- **Нативный внешний вид и ощущение.** Пользователи с трудом могут отличить нативные приложения от приложений React Native. Это происходит потому, что пользовательские интерфейсы React Native выглядят так же, как и в нативных приложениях, благодаря мощным возможностям JavaScript, который легко взаимодействует с нативной средой.
- **Большое и активное сообщество.** React Native – это платформа с открытым исходным кодом, что означает, что она доступна для всех, и любой разработчик может внести в нее свой вклад. Если вы столкнетесь с проблемой при разработке приложения, вы можете обратиться за поддержкой к сообществу.

Существуют также некоторые недостатки, о которых необходимо знать, прежде чем принимать решение о разработке приложения React Native.

- **Отладка и проблемы совместимости.** Несмотря на все замечательные возможности, React Native все еще находится в бета-версии. Поэтому у него все еще есть некоторые очевидные проблемы, такие как сложная отладка приложений и ограничения, включая проблемы совместимости.
- **Плохое управление памятью.** React Native создает приложения с большими возможностями. Однако это может быть не самая лучшая платформа для создания приложений, эффективно управляющих аппаратными ресурсами. Поскольку управление памятью в React Native не соответствует требованиям, не стоит выбирать эту платформу для создания высокопроизводительных программных решений. Если вам нужно создать приложение, которое должно выполнять сложные вычисления, лучше поискать альтернативную платформу

для разработки.

В чем различия между Flutter и React Native?



Язык программирования

Решив написать приложение на Flutter, вам придется изучить новый язык программирования – Dart. Dart относительно молод (появился в 2011 году), он был

разработан компанией Google как замена JavaScript. Это объектно-ориентированный язык с элегантным синтаксисом и хорошо документированный тысячами примеров, поэтому переход на него практически безболезненный.

React Native использует JavaScript, зрелый язык с обилием учебников и миллионами приверженцев по всему миру. С одной стороны, JavaScript уже зарекомендовал себя и сыграл ключевую роль в динамической веб-разработке. С другой стороны, Dart может удвоить производительность JavaScript благодаря опережающим и своевременным компиляторам.

Таким образом, с точки зрения скорости Flutter и Dart – очевидные лидеры; но если преобладающими критериями являются зрелость и богатый пул талантов, вам следует выбрать React Native и JavaScript.

Архитектура

При выборе кроссплатформенной среды разработки мобильных приложений важно учитывать ее техническую архитектуру. Зная внутреннюю структуру фреймворка, вы можете принять обоснованное решение и выбрать тот, который лучше всего подходит для вашего проекта.

Архитектура React Native находится под сильным влиянием паттерна JavaScript Bridge. Код, написанный на JavaScript, компилируется в машинный код во время выполнения. React Native использует архитектуру Flux, разработанную Facebook. В результате эта платформа использует специальный JavaScript-мост для связи с собственными модулями, что делает приложения немного медленнее.

Язык Dart, используемый Flutter, имеет большинство встроенных компонентов. Поэтому он больше и часто не требует места для взаимодействия с собственными модулями. Dart имеет множество фреймворков, таких как Material Design и Cupertino, которые предоставляют все необходимые технологии для разработки мобильных приложений. В свою очередь, Dart использует графический движок Skia, который написан на C++. Можно сделать вывод, что Flutter имеет все необходимые основные компоненты в самом движке, поэтому его приложения работают быстрее.

Установка

React Native можно установить с помощью менеджера пакетов Node.js NPM. Для разработчиков, имеющих опыт работы с JavaScript, процесс установки React Native

прост. Другим разработчикам потребуется изучить основы работы с менеджером пакетов NPM.

Flutter можно установить, загрузив бинарный файл для конкретной платформы с GitHub или скачав zip-архив с официального сайта. Далее необходимо добавить папку bin в переменную системного окружения. По сравнению с React Native, эта операция требует больше шагов.

Так, и React Native, и Flutter не имеют однострочной установки с помощью собственных менеджеров пакетов для конкретных ОС. Однако установка Flutter требует дополнительных шагов и поэтому является более медленным процессом.

Пользовательский интерфейс и API разработки

React Native использует нативные компоненты пользовательского интерфейса iOS и Android и предоставляет API для рендеринга пользовательских интерфейсов и доступа к устройствам. Чтобы получить доступ к большинству нативных модулей, React Native приходится полагаться на сторонние библиотеки, поэтому он сильно зависит от них.

Flutter, вместо использования нативных элементов пользовательского интерфейса, предлагает пользовательские виджеты, которые отрисовываются с нуля. Разработка пользовательского интерфейса с помощью Flutter связана с компонентами рендеринга пользовательского интерфейса, доступом к API устройств, навигацией, тестированием, управлением состояниями и множеством библиотек и предполагает обширную настройку. Такой богатый набор компонентов устраняет зависимость от сторонних библиотек. С Flutter у вас есть все необходимое для разработки мобильных приложений.

Время разработки

Программисты, имеющие опыт использования JavaScript, могут легко приступить к разработке кроссплатформенных приложений с помощью React Native. Кроме того, функция горячей перезагрузки позволяет сэкономить много времени при тестировании изменений пользовательского интерфейса. Что касается поддержки IDE, разработчики могут использовать любой текстовый редактор или среду разработки, что также помогает упростить процесс.

Flutter также имеет функцию горячей перезагрузки. Однако по мере роста

сложности приложения разработчикам необходимо изучать новые концепции Flutter, что требует дополнительного времени. Кроме того, Dart не является распространенным языком программирования, и многие IDE и текстовые редакторы не поддерживают его, что может замедлить процесс.

Возможность повторного использования кода

Хотя оба фреймворка позволяют повторно использовать код, Flutter предлагает больше возможностей в этом отношении. Во Flutter достаточно изменить всего одну строку, задать новую бизнес-логику и повторно использовать кодовую базу. В React Native возможность повторного использования не так однозначна, поскольку код не всегда совместим со всеми мобильными платформами. Разработчикам иногда приходится искать другие компоненты и корректировать кодовую базу.

Обеспечение качества

React Native – это фреймворк JavaScript, а сам JavaScript предоставляет несколько фреймворков для модульного тестирования. Однако, когда дело доходит до интеграции или тестирования на уровне пользовательского интерфейса, React Native не предоставляет никакой официальной поддержки. Существует множество сторонних инструментов, таких как Appium или Detox, которые можно использовать для тестирования приложений React Native.

Напротив, Flutter предлагает богатый набор функций для тестирования приложений на уровне модулей, виджетов и интеграций. Платформа имеет отличную документацию по тестированию приложений.

Экосистема и поддержка сообщества

React Native был запущен в 2015 году и с тех пор набирает популярность. У фреймворка есть большое сообщество разработчиков на GitHub, а также многочисленные встречи и конференции, проводимые по всему миру.

Flutter отстает на два года, но он уже привлек к себе много внимания, особенно благодаря продвижению Google. Сообщество Flutter все еще меньше по сравнению с React Native, однако в наши дни оно быстро растет, и также организуется множество встреч и конференций.

Поддержка DevOps и CI/CD

React Native не имеет официальной документации по настройке непрерывной доставки. Однако в интернете можно найти множество статей на эту тему. Так, вы можете узнать, как построить CD-конвейер приложения React Native через Azure DevOps.

Во фреймворке Flutter есть раздел о непрерывной интеграции со ссылками на внешние источники. Богатый интерфейс командной строки Flutter позволяет легко настроить непрерывную доставку. Таким образом, приложения на Flutter могут быть легко настроены на эти сервисы, в то время как React Native не предоставляет официальных инструкций по этой практике.

Релиз

Процесс выпуска приложений на Flutter стал намного проще благодаря усилиям Google. Речь идет скорее об автоматизации и гибкости. В то время как с React Native вам придется иметь дело с ручными операциями и сложными протоколами.

Производительность

Производительность – чрезвычайно важный аспект при выборе фреймворка, и в этой области Flutter выигрывает. Фреймворк ускоряет написание кода, а его движок C++ и графическая библиотека Skia позволяют создавать высокопроизводительные приложения для всех платформ.

Для связи между JavaScript и нативным кодом React Native использует мост. Это иногда задерживает рендеринг пользовательского интерфейса и приводит к другим проблемам с производительностью. Поскольку Flutter подключается к нативным элементам через встроенные библиотеки и фреймворки, ему не нужен такой механизм.

Документация

Что касается документации, то здесь Flutter также является победителем. Благодаря отличной документации новичкам легко выбрать этот фреймворк и начать его использовать. Документация по React Native намного сложнее и предполагает, что пользователь уже имеет опыт работы с JavaScript.

Заключение

И Flutter, и React Native – отличные инструменты для создания кроссплатформенных приложений. Между ними есть некоторые сходства, но есть и существенные различия, о которых вы должны знать, прежде чем решить, какой из них использовать. Дело в том, что оба инструмента являются современными, очень популярными и могут использоваться для высококлассных кроссплатформенных проектов.

Чтобы выбрать наиболее подходящий вариант, необходимо тщательно проанализировать требования к проекту и взвесить все за и против. При необходимости наши специалисты могут помочь вам в решении этой задачи.

Дата Создания
23.03.2023