Сведения об авторах

Бахтияр Муратулы Ильясов* – магистрант, Системы информационной безопасности; Казахский Национальный университет имени аль- Фараби; Республика Казахстан; e-mail: bahalar49@gmail.com.

Жанна Муратбековна Алимжанова– кандидат физико-математических наук, старший препопадаватель; Казахский Национальный университет имени аль- Фараби; Республика Казахстан.

Авторлар туралы мәліметтер

Бахтияр Мұратұлы Ильясов* – «Ақпараттық қауіпсіздік жүйелері» магистранты; әл-Фараби атындағы Қазақ ұлттық университеті; Қазақстан Республикасы; e-mail: bahalar49@gmail.com.

Жанна Мұратбекқызы Әлімжанова – физика-математика ғылымдарының кандидаты, аға оқытушы; әл-Фараби атындағы Қазақ ұлттық университеті; Қазақстан Республикасы.

Information about the authors

Bakhtiyar Muratuly Ilyassov* – Master's student, Information Security Systems; Al-Farabi Kazakh National University; The Republic of Kazakhstan; e-mail: bahalar49@gmail.com.

Zhanna Muratbekovna Alimzhanova – candidate of physical and mathematical sciences, senior teacher; Al-Farabi Kazakh National University; The Republic of Kazakhstan.

Поступила в редакцию 28.05.2024
Поступила после доработки 12.06.2024
Принята к публикации 13.06.2024

DOI: 10.53360/2788-7995-2024-2(14)-4

МРНТИ: 49.38.49

А.А. Мухамедин*, Г.А. Абитова
Astana IT University
010000, Республика Казахстан, г. Астана, пр. Мангилик Ел, 55/11
*e-mail: 222153@astanait.edu.kz

ИССЛЕДОВАНИЕ REACT NATIVE И FLUTTER, КРОССПЛАТФОРМЕННЫХ ФРЕЙМВОРКОВ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Аннотация: *В статье выявлено, что создание мобильных приложений отдельно для Android и iOS становится сложным и затратным процессом. Появляется необходимость в общем решении, которое упростит процесс разработки, поддержки, тестирования и развертывания на различных платформах. Это решение должно стандартизировать процесс создания мобильных приложений.*

React Native, созданный Facebook, представляет собой значимый этап в развитии кроссплатформенной разработки мобильных приложений. Благодаря активному и мощному сообществу, React Native стал самым востребованным инструментом для создания кроссплатформенных приложений. Тем не менее, Google решил разработать свое собственное решение, Flutter, после тщательного анализа преимуществ и недостатков React Native. Flutter нацелен на оптимизацию для мобильных устройств и стремится предоставить разработчикам полноценное и окончательное решение для создания кроссплатформенных приложений.

В данной статье анализируются ключевые характеристики React Native и Flutter, проводится исследование и сравнение этих характеристик с целью выявления причин их различий. Авторы статьи надеются, что результаты исследования помогут совершенствовать кроссплатформенную разработку и обеспечат дальнейший прогресс в этой области.

Ключевые слова: *Andorid, фильмы, разработка, искусственный интеллект, Архитектура, кроссплатформенный, исследование, интеграция, IOS, React Native, Flutter.*

Введение

С каждым днем мобильные приложения становятся все более существенными для нашей повседневной жизни. Согласно данным с ноября 2016 года, объем сетевого трафика, который передается через мобильные устройства, составляет 48,19%, что превышает объем трафика от настольных компьютеров и ноутбуков (47%) [1].

Для того чтобы привлечь максимальное число пользователей, мобильное приложение должно быть доступно как на платформе Android, так и на iOS. Однако различия между этими двумя платформами значительны, и часто требуют от разработчиков различных навыков: например, для Android нужно знание Java или Kotlin, а для iOS – Objective-C или Swift. Это создает сложности для разработчиков и компаний, которые стремятся создать кроссплатформенные приложения.

Facebook представил React Native в марте 2015 года как открытую кроссплатформенную среду JavaScript, которая призвана решить проблему разработки приложений для разных платформ. Этот фреймворк основан на платформе React, которую Facebook представил несколько лет ранее [2]. React широко популярен среди разработчиков благодаря своей простоте, легкости и эффективному процессу разработки [3].

В конце 2016 года Google представила новый мобильный инструмент разработки под названием Flutter. Этот фреймворк, вдохновленный React Native, также позволяет создавать приложения для обеих платформ, что упрощает и удешевляет процесс разработки для iOS и Android. В отличие от React Native, Flutter был разработан с нуля. На момент написания данного исследования (август 2017 года) использование Flutter в коммерческих проектах ограничивалось только Google.

Различные компании уже экспериментировали с кроссплатформенными фреймворками, такими как React Native и Flutter, но до сих пор ни один из них не смог полностью удовлетворить требования промышленного развития. Несмотря на неудачи предыдущих попыток, интерес к React Native и Flutter с поддержкой Facebook и Google остается высоким, и люди с оптимизмом смотрят на их перспективы.

Эта работа направлена на проведение всестороннего анализа React Native и Flutter. В ней будут рассмотрены и проиллюстрированы основные концепции и характеристики обеих платформ, а также будет проведено сравнение их производительности и процесса разработки [10].

Статья будет посвящена анализу React Native и Flutter. Кроме того, чтобы выявить различия между ними, будет выполнена полная перепись рабочего приложения React Native с использованием Flutter в качестве альтернативы.

Теоретическая основа

Сообщество React Native считается одним из самых мощных в мире открытого исходного кода. На самом деле, на данный момент (март 2018 года), React Native занимает третье место по популярности среди проектов на GitHub. Значительный вклад в его развитие вносят не только отдельные разработчики и Facebook, но и такие крупные технологические компании, как Microsoft и Samsung [5].

Одной из наиболее увлекательных особенностей React Native является его способность переносить современные веб-технологии на мобильные устройства, сохраняя при этом функциональность и производительность на высоком уровне. Несмотря на то, что приложения React Native в основном разрабатываются на JavaScript и исполняются в контексте JavaScript, это не делает их гибридными или основанными на технологии HTML5. За счет использования нативных компонентов интерфейса приложения React Native имеет возможность отображать элементы пользовательского интерфейса и взаимодействовать с аппаратным обеспечением устройства, таким как камера и хранилище.

JSX:

При изучении кода приложения на React Native одним из простейших наблюдений является использование JSX. JSX – это специальное расширение синтаксиса JavaScript, предназначенное в основном для описания внешнего вида пользовательского интерфейса. При компиляции приложения JSX преобразуется в обычные объекты JavaScript. В приведенном примере создается компонент, содержащий две кнопки, каждая из которых выполняет различные действия. Функция render() имеет ключевое значение для каждого компонента React, поскольку она возвращает объект представления компонента. Фигурные скобки используются для вставки свойств компонента в объект представления. Применение

стрелочной функции в action1() и action2() демонстрирует использование современных возможностей JavaScript в JSX.

Поскольку JSX используется для описания пользовательского интерфейса, может показаться, что это просто еще один язык разметки, подобный HTML или XAML. Однако это неверно. JSX и обычный JavaScript объект могут взаимодействовать друг с другом, что позволяет вставлять обычные выражения JavaScript прямо внутри JSX кода.

Важно отметить, что использование JSX в React позволяет предотвратить атаки через внедрение. React DOM автоматически преобразует любые введенные значения в обычные строки перед их отображением. Это означает, что пользователь не может внедрить какие-либо скрипты или команды в ваше приложение через его интерфейс [4].

Virtual DOM:

Использование виртуального DOM является одной из ключевых причин того, что приложение React Native может работать на различных платформах. Виртуальный DOM позволяет React манипулировать упрощенным деревом DOM, которое соответствует реальному дереву DOM, для оптимизации производительности [12].

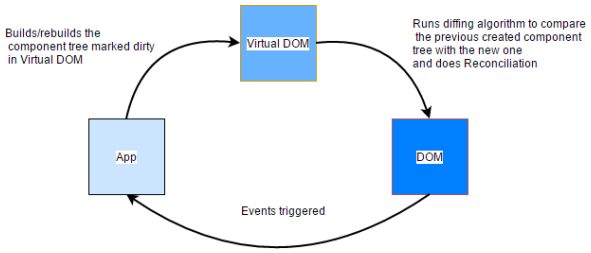

Рисунок 1 – Рабочий процесс Virtual DOM

На рисунке 1 изображены взаимосвязи между приложением, DOM и Virtual DOM. Любое действие пользователя в DOM, такое как нажатие кнопки, в конечном итоге порождает события в приложении, которое затем определяет структуру Virtual DOM. Приложение также периодически запускает алгоритм сравнения для эффективного обновления Real DOM.

В React каждый раз, когда требуется отобразить элемент JSX, соответствующий ему виртуальный объект DOM обновляется одновременно. Перед обновлением React создает снимок текущего дерева Virtual DOM. По завершении обновления React сравнивает обновленное дерево DOM с предыдущим снимком, чтобы определить точные части, которые необходимо повторно отобразить в реальном дереве DOM. Этот процесс, известный в мире React как «diffing», требует использования набора алгоритмов.

Помимо эффективного алгоритма сравнения, React прикладывает значительные усилия для пакетной обработки операций чтения/записи DOM. Находя минимальное количество шагов для обновления виртуального DOM, React выполняет все эти операции в рамках одного цикла событий, не затрагивая реальный DOM. Только после завершения цикла событий React перерисовывает настоящий DOM. Таким образом, существует только один момент, когда необходима перерисовка настоящего DOM [6].

Сочетание процесса «diffing» и определения, какие узлы в дереве представления необходимо обновить, называется согласованием. В React согласование и рендеринг являются двумя отдельными этапами. Поэтому React и React Native могут использовать свои собственные методы рендеринга, при этом воспользовавшись одним и тем же примирителем, предоставляемым ядром React.

Flutter:

В процессе изучения кода приложения React Native один из самых простых аспектов, который бросается в глаза – это использование JSX. JSX представляет собой специальное расширение синтаксиса JavaScript, которое в основном применяется для определения того, как должен быть отображен пользовательский интерфейс.

Как и в предыдущей части, мы рассмотрим некоторые ключевые аспекты Flutter. Наша цель здесь – понять основной принцип работы Flutter и ознакомиться с циклом его разработки.

Flutter – это кроссплатформенный фреймворк, созданный с целью разработки высокопроизводительных мобильных приложений. Google выпустил Flutter публично в 2016

году. Приложения, разработанные на Flutter, могут быть запущены не только на Android и iOS, но и на Fuchsia – операционной системе нового поколения от Google, которая выбирает Flutter в качестве своей основной платформы приложений.

Flutter является уникальным в своем роде. В отличие от использования веб-представлений или OEM-виджетов устройств, Flutter визуализирует каждый компонент пользовательского интерфейса с помощью собственного высокопроизводительного механизма рендеринга. Такой подход обеспечивает возможность создания приложений с высокой производительностью, сравнимой с нативными приложениями. С точки зрения архитектуры, код движка на C/C++ компилируется с использованием Android NDK и LLVM на iOS, в то время как любой код на Dart компилируется с помощью AOT.

Важной особенностью Flutter является поддержка горячей перезагрузки с сохранением состояния во время разработки, что значительно ускоряет цикл разработки. Горячая перезагрузка с сохранением состояния позволяет внедрять обновленный исходный код в работающую виртуальную машину Dart без изменения внутренней структуры приложения. Это означает, что все текущие переходы и действия в приложении сохраняются после горячей перезагрузки, что значительно упрощает и ускоряет процесс разработки [14].

Сравнение производительности

Существует множество различий в реализации Flutter и React Native, и поэтому эффективность одного и того же сценария может значительно различаться. Технически Flutter благодаря своей архитектуре снизу-вверх обычно более эффективен и менее требователен к ресурсам. Однако у React Native существует более крупное сообщество разработчиков. Например, зрелые библиотеки, такие как Redux, которые широко используются в тематических исследованиях, изначально разработаны для React. В общем, нелегко сделать объективное заключение относительно эффективности без учета конкретного контекста каждого случая.

Для измерения производительности важно понимать некоторые основные концепции. Один из таких концептов - кадры в секунду (FPS), который является одним из самых простых и широко используемых показателей для оценки плавности работы приложения. Каждый кадр представляет собой статическое изображение текущего состояния окна. Любые даже незначительные изменения в текущем кадре приводят к созданию нового кадра. Записывая количество кадров, отображаемых каждую секунду на определенном устройстве, можно легко сравнить производительность двух различных приложений.

Цель этого раздела – продемонстрировать несовершенное сравнение производительности приложений Flutter и React Native. Для этого в качестве тестовых примеров выбраны два наиболее часто встречающихся сценария мобильного приложения. Все тестирование и сравнения будут проводиться на одном устройстве с Android (Oneplus A3003) для мониторинга частоты кадров [11].

Scroll:

Вертикальный прокручиваемый список – это типичное представление во всех мобильных приложениях. Это настолько распространено, что как Android, так и iOS предоставляют специальные компоненты представлений (RecyclerView и UICollectionView), которые специально оптимизированы для обеспечения высокой производительности при работе с большими наборами данных. Прокрутка, будучи простым действием, требует мгновенной обратной связи, анимации и предварительного рендеринга, чтобы обеспечить плавный пользовательский опыт [9].

В марте 2017 года React Native официально прекратил поддержку компонента ListView и вместо него представил новый компонент FlatList в качестве основного инструмента для создания прокручиваемых списков. Новый FlatList разработан для оптимизации использования памяти и предоставляет современные функции, такие как Pull to Refresh, при этом имея более упрощенный API. С другой стороны, в Flutter компонент ListView обновлялся не так часто. Он был создан как специализированный дочерний виджет пользовательского ScrollView для отображения набора данных в линейном порядке [8].

Для минимизации влияния других факторов был создан чрезвычайно упрощенный пример, включающий только компонент или виджет фреймворка. Суть примера заключается в вертикальном списке прокрутки с 1000 элементами. Каждый элемент содержит изображение и две строки текста. На рисунке ниже представлены некоторые из наиболее важных показателей производительности [7].

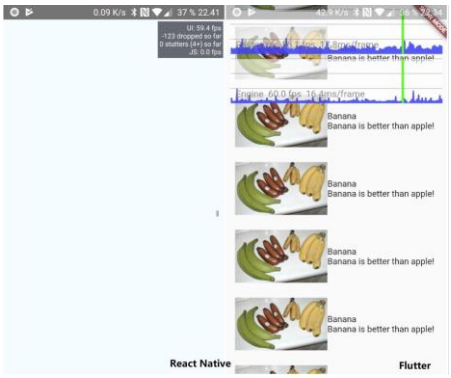

Рисунок 2 – Монитор FPS на устройстве

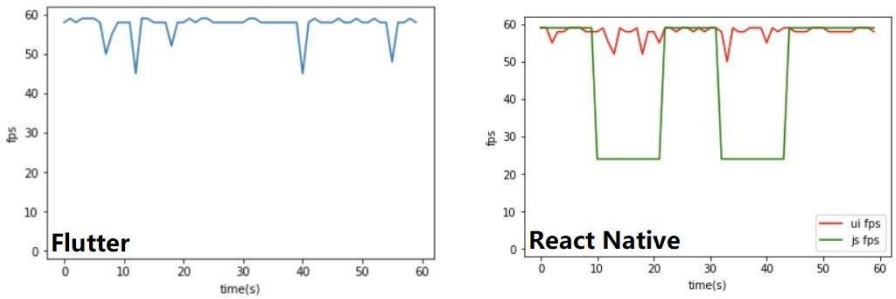

Рисунок 3 – Отслеживание FPS при прокрутке

Как показано на рисунках 2 и 3, как Flutter, так и React Native успешно справляются с прокруткой, и средний FPS во время прокрутки всегда оставался выше 60. Однако показатель FPS в потоке пользовательского интерфейса не полностью отражает производительность React Native. Как было упомянуто в предыдущем разделе, большая часть приложений React Native выполняется в потоке JavaScript. Более серьезной проблемой является то, как FlatList в React Native оптимизирует использование памяти. Для экономии памяти FlatList отображает только определенное количество элементов одновременно для плотного списка с большим количеством элементов. В определенных сценариях, например, при быстрой прокрутке, пустые блоки могут отображаться как заполнители, что приводит к снижению частоты кадров в потоке JavaScript. С другой стороны, FPS в Flutter остается стабильным. Пики на графике указывают на то, что совместная обработка пользовательского ввода и рендеринга анимаций требует дополнительных ресурсов.

Disk I/O:

I/O скорость ввода-вывода (I/O) играет критическую роль в производительности, особенно когда речь идет о мобильных приложениях. Это относится к скорости обмена данными между приложением и хост-устройством, таким как чтение и запись файлов на диске. Такие операции часто встречаются при взаимодействии с хранилищем данных устройства.

Операции с системными файлами в React Native обычно выполняются с помощью библиотеки «react-native-fs». Эта библиотека предоставляет доступ к файловой системе устройства и предлагает простой API для асинхронного чтения и записи файлов. В сообществе Flutter также существует аналогичный плагин под названием «path_provider», который обеспечивает доступ к файловой системе хост-устройства и предоставляет необходимые функции для работы с файлами. Обе эти библиотеки используют встроенные оптимизации файловой системы для улучшения производительности операций с файлами.

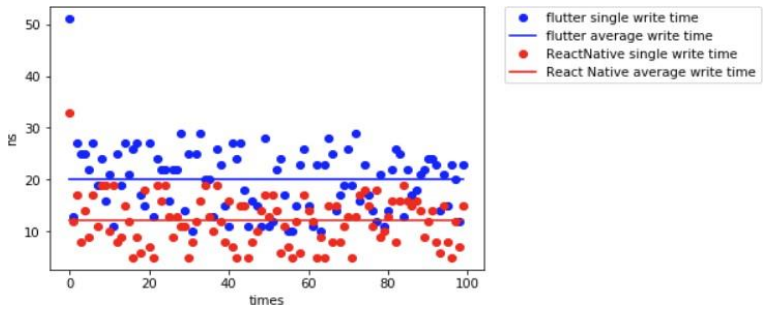

Рисунок 4 – Написание файлов Char в Flutter (синий) и React Native (красный) требует времени

Как показано на рисунке 4, React Native демонстрирует преимущество как по среднему времени (прямая линия), так и по единичному времени (точки). Это обусловлено процессом оптимизации, реализованным в библиотеке «react-native-fs». Фактически, в отношении скорости записи React Native достигает производительности, сравнимой с нативным приложением. Еще одним интересным наблюдением является то, что как Flutter, так и React Native требуют относительно много времени на инициализацию при первоначальной записи. Возникает предположение, что инициализация экземпляра собственного ввода-вывода файлов является предварительным условием для достижения высокой скорости записи.

Заключение

Цель данной статьи заключается в проведении всестороннего сравнительного исследования между React Native и Flutter. В ходе исследования рассмотрены и представлены теоретические основы обеих платформ, а также их основные характеристики. Для сравнения процессов разработки была успешно переписана демонстрационная программа React Native с открытым исходным кодом на Flutter. Были проведены и анализированы различные тесты, касающиеся производительности приложений.

Однако, из-за ограниченных временных и ресурсных возможностей не все аспекты платформ были рассмотрены в подробностях. Например, не был подробно обсужден процесс рендеринга, на оптимизацию которого были направлены значительные усилия как в React Native, так и в Flutter. Кроме того, сравнение нативных и кросс-платформенных приложений не было исчерпывающе рассмотрено.

Работа React Native во многом оказала значительное влияние на последующие разработки. Инновационные концепции React, такие как однонаправленный поток данных и JSX, успешно воплощены и переосмыслены в React Native. Благодаря мощному сообществу, React Native является одним из лучших выборов для разработки кроссплатформенных приложений с нуля.

У Flutter светлое будущее. Он успешно сохраняет сложный дизайн React Native и даже улучшает его собственные разработки. Основное внимание уделяется согласованности и ясности синтаксиса, а также высокому уровню SDK, что действительно радует разработчиков. Благодаря рендерингу виджетов с помощью специализированного движка улучшается производительность и исключаются возможные проблемы с OEM-производителями.

В заключение можно сказать, что и React Native, и Flutter подтвердили свою ценность как кроссплатформенные среды разработки мобильных приложений. Преимущества в эффективности и удобстве разработки могут значительно ускорить процесс внедрения продукта на рынок. Теперь создание качественного и привлекательного приложения для всех основных мобильных платформ стало более доступным. При этом важно учитывать, что некоторое снижение производительности по сравнению с нативным приложением может быть оправданным компромиссом.

Список литературы

1. Griffiths D. React Native: Mobile App Development with JavaScript / D. Griffiths, R. Fruechte, E. Goldberg. – 2018.
2. Абрамов Д. Redux официальное руководство / Д. Абрамов. – 2020.
3. Pratt D. React Native From Zero to One / D. Pratt. – 2021.

4. Краснов А. Разработка мобильных приложений на платформе Android / А. Краснов, В. Арутюнов, В. Данильченко. – 2020.
5. Дубровский И. Разработка мобильных приложений на Flutter: создание iOS и Android приложений на одном коде / И. Дубровский, С. Быков. – 2019.
6. Ryan R. Designing Mobile Interfaces: Patterns for Interaction Design / R. Ryan. – 2019.
7. React Native Documentation [online resource] / URL: https://reactnative.dev/
8. Redux Documentation [online resource] / URL: https://redux.js.org/
9. Flutter Documentation [online resource] / URL: https://flutter.dev/
10. ChatGPT API Documentation [online resource] / URL: https://beta.openai.com/docs/
11. Fowler M. Clean Code: A Handbook of Agile Software Craftsmanship / M. Fowler. – 2008.
12. Kernighan B. The C Programming Language / B. Kernighan, D. Ritchie – 1988.

References

1. Griffiths D. React Native: Mobile App Development with JavaScript / D. Griffiths, R. Fruechte, E. Goldberg. – 2018. (In English).
2. Abramov D. Redux ofitsial'noe rukovodstvo / D. Abramov. – 2020. (In Russian).
3. Pratt D. React Native From Zero to One / D. Pratt. – 2021. (In English).
4. Krasnov A. Razrabotka mobil'nykh prilozhenii na platforme Android / A. Krasnov, V. Arutyunov, V. Danil'chenko. – 2020. (In Russian).
5. Dubrovskii I. Razrabotka mobil'nykh prilozhenii na Flutter: sozdanie iOS i Android prilozhenii na odnom kode / I. Dubrovskii, S. Bykov. – 2019. (In Russian).
6. Ryan R. Designing Mobile Interfaces: Patterns for Interaction Design / R. Ryan. – 2019. (In English).
7. React Native Documentation [online resource] / URL: https://reactnative.dev/. (In English).
8. Redux Documentation [online resource] / URL: https://redux.js.org/. (In English).
9. Flutter Documentation [online resource] / URL: https://flutter.dev/. (In English).
10. ChatGPT API Documentation [online resource] / URL: https://beta.openai.com/docs/. (In English).
11. Fowler M. Clean Code: A Handbook of Agile Software Craftsmanship / M. Fowler. – 2008. (In English).
12. Kernighan B. The C Programming Language / B. Kernighan, D. Ritchie – 1988. (In English).

А.А. Мухамедин*, Г.А. Абитова
Astana IT University,
010000, Қазақстан Республикасы, Астана қ., Мәңгілік Ел даңғылы, 55/11
*e-mail: 222153@astanait.edu.kz

REACT NATIVE ЖӘНЕ FLUTTER, КРОСС-ПЛАТФОРМАЛЫҚ МОБИЛЬДІ ҚОСЫМШАЛАР ШЕҢБЕРЛЕРІН ЗЕРТТЕУ

Мақалада Android және iOS үшін мобильді қосымшаларды бөлек жасау күрделі және қымбат процесске айналып бара жатқанын көрсетеді. Әртүрлі платформаларда әзірлеу, қолдау, тестілеу және орналастыру процесін жеңілдететін ортақ шешім қажет. Бұл шешім мобильді қосымшаларды жасау процесін стандарттауы керек.

Facebook жасаған React Native кросс-платформалық мобильді қосымшаларды әзірлеудегі маңызды кезең болып табылады. Белсенді және қуатты қауымдастықтың арқасында React Native кросс-платформалық қосымшаларды жасаудың ең танымал құралы болды. Дегенмен, Google React Native-тің артықшылықтары мен кемшіліктерін мұқият талдағаннан кейін Flutter атты жеке шешімін әзірлеуге шешім қабылдады. Flutter мобильді құрылғыларды оңтайландыруға бағытталған және әзірлеушілерге кросс-платформалық қосымшаларды жасау үшін толық және түпкілікті шешім беруге тырысады.

Бұл мақалада React Native және Flutter негізгі сипаттамалары талданады, олардың айырмашылықтарының себептерін анықтау үшін осы сипаттамалар зерттеледі және салыстырылады. Мақала авторлары зерттеу нәтижелері кросс-платформалық дамуды жақсартуға және осы саладағы одан әрі ілгерілеуді қамтамасыз етуге көмектеседі деп үміттенеді.

Түйін сөздер: *Andorid, фильмдер, әзірлеу, жасанды интеллект, архитектура, кросс-платформа, зерттеу, интеграция, IOS, React Native, Flutter.*

A.A. Mukhamedin*, G.A. Abitova
Astana IT University,
010000, Республика Казахстан, г. Астана, проспект Мангилик Ел, 55/11
*e-mail: 222153@astanait.edu.kz

RESEARCH OF A REACT NATIVE VS FLUTTER, CROSS-PLATFORM MOBILE APPLICATION FRAMEWORKS

The article reveals that creating mobile applications separately for Android and iOS is becoming a complex and costly process. There is a need for a common solution that will simplify the process of development, support, testing and deployment on various platforms. This solution should standardize the process of creating mobile applications.

React Native, created by Facebook, represents a significant milestone in the development of cross-platform mobile application development. Thanks to an active and powerful community, React Native has become the most popular tool for creating cross-platform applications. However, Google decided to develop its own solution, Flutter, after carefully analyzing the advantages and disadvantages of React Native. Flutter is focused on optimizing for mobile devices and strives to provide developers with a complete and definitive solution for creating cross-platform applications.

This article analyzes the key characteristics of React Native and Flutter, researching and comparing these characteristics in order to identify the reasons for their differences. The authors of the article hope that the results of the study will help improve cross-platform development and ensure further progress in this area.

Key words: *Andorid, movies, development, artificial intelligence, Architecture, cross-platform, research, integration, IOS, React Native, Flutter.*

Сведения об авторах

Айдар Алибекулы Мухамедин* – магистрант Astana IT University, Республика Казахстан, г.Астана; e-mail: 222153@astanait.edu.kz

Гулнара Аскеровна Абитова – научный руководитель, PhD, доцент, Astana IT University; Республика Казахстан, г. Астана; e-mail: gulnara.abitova@astanait.edu.kz

Авторлар туралы мәліметтер

Айдар Алибекулы Мухамедин* – магистрант, Astana IT University; Қазақстан Республикасы, Астана қ.; e-mail: 222153@astanait.edu.kz

Гүлнара Әскерқызы Әбитова – PhD, доцент; Astana IT University; Қазақстан Республикасы, Астана қ.; e-mail: gulnara.abitova@astanait.edu.kz

Information about the authors

Aidar Alibekuly Mukhamedin – Master degree, Astana IT University, Republic of Kazakhstan, Astana; e-mail: 222153@astanait.edu.kz

Gulnara Askerovna Abitova – scientific advisor, PhD, Associate Professor, DIS&CS, Astana IT University; Republic of Kazakhstan, Astana; e-mail: gulnara.abitova@astanait.edu.kz

Поступила в редакцию 08.04.2024
Принята к публикации 15.05.2024

DOI: 10.53360/2788-7995-2024-2(14)-5

IRSTI: 50.47.29

Y. Ospanov*, N. Turarbai[1], M. Bayraktar[2]
[1]Shakarim University of Semey,
071412, Republic of Kazakhstan, Semey, 20 A Glinka Street
[2]Akdeniz University,
Pınarbası Mah. Dumlupinar Boulevard 07070 Campus Konyaalti, Antalya, Turkey
*e-mail: 78oea@mail.ru

FUZZY DECISION-MAKING PROBLEMS FOR CONTROLLING OPERATING MODES OF TECHNOLOGICAL SYSTEMS AND METHODS FOR SOLVING THEM

Abstract: *Statements of decision-making problems for the control of fuzzy technological objects are formalized and obtained, and methods for solving them are proposed. The object of study is the heating stations of the «hot» main oil pipeline. Since such objects are often characterized by multicriteria and often operate in conditions of unclear initial information, the tasks are formalized in the form of multicriteria decision-making problems in a fuzzy environment. Based on the modification of various optimality principles, new mathematical formulations of the problems to be solved were obtained and interactive heuristic algorithms for solving them were developed. The novelty of the proposed approaches to solving formalized fuzzy problems*