Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра информационной безопасности

Багров Алексей Сергеевич

Сравнительный анализ фреймворков мобильной разработки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

с.н.с., д.т.н.

Намиот Дмитрий Евгеньевич

Москва, 2023

Аннотация

В данной работе рассматриваются четыре фреймворка мобильной разработки: Flutter, React Native, Apache Cordova и Angular.

А также разрабатываются критерии оценки и сравнения данных средств кроссплатформенной разработки, с целью выбора конкретного инструмента для создания конкретного программного продукта. Иными словами руководство по выбору необходимого средства разработки.

Содержание

1. Введение

Предметом изучения данной работы являются различные средства кроссплатформенной разработки, такие как Flutter, React Native, Apache Cordova и Angular. Широкий спектр задач и, соответственно, специфика и способы их решения определяют то, каким именно средством разработки следует пользоваться для решения каждой конкретной задачи.

Можно по-разному подходить к решению задачи и избирать различные пути её решения. Но от корректного (с точки зрения прагматичности и удобства программирования) выбора инструмента разработки зависит потенциально возможный реализуемый функционал, качество итогового продукта и время, которое будет затрачено на решение задачи.

В ходе данной работы рассматривается и проводится сравнительный анализ нескольких средств разработки. А также разрабатывается руководство (критерии) по выбору необходимого средства разработки программного продукта. Существует много факторов, от которых зависит этот выбор. Данная работа посвящена выявлению и исследованию этих факторов.

2. Постановка цели и задачи

2.1. Цели работы

- разработка руководства по выбору средства кроссплатформенной разработки

2.2. Задачи работы

- рассказать про каждое из средств кроссплатформенной разработки
- выделить плюсы и минусы каждого инструмента
- выделить критерии сравнения средств кроссплатформенной разработки
- сравнить работу инструментов на одинаковых задачах (анализ программ, написанных с помощью каждого из инструмента кроссплатформенной разработки)

3. Рассматриваемые средства разработки

Прежде всего, стоит сказать, что разработку мобильных приложений можно разделить на нативную и кроссплатформенную.

Кроссплатформенность — это способность программного обеспечения работать с несколькими аппаратными платформами или операционными системами. Кроссплатформенность обеспечивается благодаря использованию высокоуровневых языков программирования, сред разработки, сред выполнения, поддерживающих условную компиляцию, компоновку и выполнение кода для различных платформ. Примером служит программное обеспечение, предназначенное для работы в операционных системах Linux и Windows одновременно или же IOS и Android в случае мобильной разработки.

Кроссплатформенные инструменты мобильной разработки — это программные платформы, позволяющие разработчикам создавать мобильные приложения, которые могут работать на нескольких операционных системах. Эти инструменты позволяют разработчикам единожды писать код (**единая код-база**) и «развернуть» его на нескольких мобильных платформах, таких как Android, iOS, тем самым, сокращая время, усилия и затраты, необходимые для разработки отдельных приложений для каждой платформы.

В нативных приложениях код создаётся с нуля под каждую платформу, на языке данной конкретной платформы, со стандартными библиотеками и SDK (software development kit). Например, для Android используются Java или Kotlin. Для iOS используются языки программирования Objective-C или Swift. В кроссплатформенных приложениях код пишется один раз и затем компилируется для iOS, Android и других платформ. Приложения будут работать и выглядеть в нескольких мобильных операционных системах идентично.

В настоящее время выделяют несколько наиболее распространённых и удобных средств для кроссплатформенной разработки, которые призваны частично или целиком заменить собой нативную разработку в тех случаях, когда это является действительно необходимым. Рассмотрим их.

3.1. Flutter

Flutter [1] — это комплект средств разработки и кроссплатформенная платформа с открытым исходным кодом для создания, как высокопроизводительных мобильных прило-

жений под операционные системы Android и iOS, так и веб-приложений с использованием языка программирования Dart [2]. Также есть возможность создания настольных (desktop) приложений для Windows, macOS, Linux и Google Fuchsia. Фреймворк Flutter разработан и развивается в настоящее время корпорацией Google.

В основе фреймворка Flutter лежит мультипарадигменный язык программирования Dart, также созданный компанией Google. Этот язык позиционируется в качестве замены (альтернативы) языку программирования JavaScript. Многими членами программистского сообщества считается, что JavaScript «имеет фундаментальные изъяны», которые уже невозможно исправить. Под данным тезисом и был создан Dart. Код на Dart компилируется в нативный код. Это позволяет повысить производительность приложений и ускорить время разработки.

Основная идея и цель создания Flutter — не просто создание фреймворка исключительно для мобильных приложений, а разработка кроссплатформенного именно UI-фреймворка, что позволяет создавать как UI, так и приложения с нуля целиком.

Основные составляющие комплекта разработки Flutter

1. платформа Dart
2. движок Flutter
3. библиотека Foundation (основные методы и классы для работы; написана на языке Dart)
4. наборы виджетов
5. средства разработки (Flutter DevTools)

В основе разработки приложений на Flutter стоят виджеты. Виджеты — это изменяемые объекты какой-либо части пользовательского интерфейса. Все графические объекты (суть, дизайн приложения, пользовательский интерфейс), включая текст, всевозможные кнопки, всплывающие окна, формы, анимацию и сенсорное взаимодействие, создаются с помощью виджетов.

Виджеты описывают, каким будет их вид в зависимости от конфигурации, положения и состояния. Когда состояние виджета изменяется (например, пользователь нажимает

кнопку), виджет перестраивает своё описание, и его внешний вид меняется. Комбинированием простых виджетов создаются сложные виджеты. Благодаря наличию виджетов приложение на Flutter выглядит естественно и работает на различных устройствах с различными ОС. Это позволяет легче создавать сложные элементы пользовательского интерфейса и легче настраивать его. Но приложения можно создавать и не прибегая к виджетам, а именно, напрямую вызывая методы библиотеки Foundation.

За счёт всего этого достигаются быстрый, плавный и отзывчивый пользовательский интерфейс, анимация, графика и переходы.

3.2. React Native

React Native [3] — это кроссплатформенный фреймворк с открытым исходным кодом для разработки мобильных приложений, а также настольных (desktop) приложений на языке JavaScript. Данный фреймворк создан компанией Facebook. Платформы, поддерживаемые React Native: Android, iOS, macOS, Web, Windows, UWP (Universal Windows Platform).

React Native — это "компонентный фреймворк что означает, что все в React Native является компонентом. Это облегчает повторное использование кода и создание сложных элементов пользовательского интерфейса.

Основные принципы работы React Native практически идентичны принципам работы библиотеки React [1], за исключением того, что React Native управляет не браузерной DOM, а платформенными интерфейсными компонентами. JavaScript-код, написанный разработчиком, выполняется в фоновом потоке, и взаимодействует с платформенными API через асинхронную систему обмена данными, называемую «Bridge».

Система стилей (способ конфигурации визуальных свойств элементов интерфейса) React Native имеет синтаксис, похожий на CSS. Но при этом фреймворк не использует технологии HTML или CSS. Вместо этого для каждой из поддерживаемых фреймворком операционных систем реализованы программные адаптеры, применяющие заданный разработчиком стиль к платформенному интерфейсному элементу (таким образом, как и в

[1]это JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов, основной идеей которой является организация высокой скорости разработки, простоты и масштабируемости

случае с Flutter, приложения будут работать и выглядеть в разных мобильных операционных системах идентично).

Ещё одно очень важное свойство фреймворка React Native является возможность использовать уже существующий код, написанный на других языках программирования (например, Java или Kotlin для Android и Objective-C или Swift для iOS). React Native поддерживает интеграцию в уже существующие приложения. Таким образом, часть интерфейса мобильного приложения может быть реализована на React Native, а остальная часть — при помощи чисто нативных платформенных средств. Это бывает полезно в тех ситуациях, когда нецелесообразно переписывать целиком программный продукт, а нужно лишь добавить в него некоторый новый функционал, который по разным оценкам может быть эффективнее реализовать с помощью фреймворка.

3.3. Apache Cordova

Apache Cordova (ранее «PhoneGap») [4] — мобильная среда разработки приложений, фреймворк с открытым исходным кодом. Первоначально разработана компанией Nitobi в 2009 году. Фреймворк написан на языках JavaScript, Java, C++, C#, Objective-C, Node.js. В последствии фреймворк был приобретен компанией Adobe Systems в 2011 году, которая произвела ребрендинг «PhoneGap» и продолжила развивать данную технологию. В итоге была выпущена версия с открытым исходным кодом программного обеспечения под названием Apache Cordova.

Apache Cordova позволяет программистам создавать приложения для мобильных устройств с помощью таких веб-технологий, как CSS3, HTML5 и JavaScript вместо того, чтобы использовать конкретные платформы API (для Android, IOS или Windows Phone). Это обеспечивается за счёт преобразования кода из CSS, HTML, JavaScript в код, который любая платформа воспринимает как web-элемент. Это расширяет возможности HTML и JavaScript для работы с различными устройствами. В результате приложения являются своего рода «гибридными», то есть они не являются ни по-настоящему мобильными приложениями (ведь генерация макета осуществляются с помощью web-view вместо основной структуры пользовательского интерфейса платформы), ни web-приложениями (потому что они являются не только своего рода web-приложениями, но и упакованы они в качестве приложений для распределения, к тому же они имеют доступ к API базового

функционала устройства).

Основное отличие в разных версиях данной технологии заключается в том, что ранним версиям «PhoneGap» требовался компьютер от Apple, чтобы создавать приложения под iOS, и компьютер с Windows, чтобы создавать приложения под Windows. После же ребрендинга компанией Adobe Systems стало возможным подгружать исходный код CSS, HTML, JavaScript в «облачный компилятор», который собирает приложения под каждую поддерживаемую платформу.

3.4. Angular

Angular [5] — это среда проектирования приложений, открытая и свободная платформа для разработки веб-приложений (в том числе и одностраничных [2]), написанная на языке TypeScript (типизированная версия JavaScript), разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний. Angular — полностью переписанный фреймворк от той же команды, которая написала AngularJS. Изначально данный фреймворк создавался как вторая версия AngularJS. Angular2, переписанный с нуля на TypeScript, обладает иной архитектурой и не является обратно совместимым с AngularJS, в связи с чем для предотвращения путаницы было решено развивать его как отдельный фреймворк, нумерация версий которого начинается с "2".

Angular включает в себя фреймворк для создания масштабируемых веб-приложений, коллекцию интегрированных библиотек, которые охватывают широкий спектр функций (включая маршрутизацию, управление формами, взаимодействие клиент-сервер), набор инструментов для разработки, сбора и тестирования кода.

Angular имеет мощный интерфейс командной строки (CLI), который облегчает генерацию кода, сборку, тестирование и развертывание приложений.

[2] это веб-приложение или веб-сайт, использующий единственный HTML-документ как оболочку для всех веб-страниц и организующий взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript, обычно посредством AJAX (если более конкретно, то AJAX — это подход к построению интерактивных пользовательских интерфейсов веб-приложений; AJAX служит для динамического изменения содержания страницы без необходимости перезагрузки всей страницы полностью, следовательно веб-приложения становятся быстрее и удобнее за счёт уменьшения нагрузки на сервер)

3.5. Промежуточный вывод

Каждый представленный выше инструмент имеет свои преимущества и недостатки, поэтому необходимо определить свои конкретные потребности и выбрать тот инструмент, который наилучшим образом соответствует требованиям и задачам, которые стоят перед аналитиками и разработчиками.

4. Выбор между нативной и кроссплатформенной разработкой

Когда же всё-таки и почему стоит сделать выбор в пользу кроссплатформенной разработки. А когда лучше в пользу нативной?

Кроссплатформенная разработка подходит, например, для тех ситуаций, когда программный продукт ещё не зарекомендовал себя, пользовательский спрос не сформирован и его необходимо быстро вывести на рынок в качестве "пилотной версии". Или же в той ситуации, когда рынок сформирован, но с точки зрения бизнеса ставится задача обогнать конкурентов и реализовать новую функциональность — в данном случае важна высокая скорость разработки сразу на две платформы, Android и iOS. Соответственно, не нужно тратить много времени на написание кода, долгое тестирование, тем самым можно сэкономить ресурсы компании, временные, трудовые и финансовые.

Нативная же разработка больше подходит мультифункциональным проектам с высокой нагрузкой, многоступенчатой вложенностью и "сложной логикой таким как сложные высоконагруженные приложения банков, где в приложении собрано множество сервисов, что образует некую экосистему, или крупным торговым площадкам с большим количеством контента. В данном случае в работу приложений интегрирована огромная база данных, а само приложение единовременно обрабатывает большое число запросов от клиентов. Также, если в компании уже сформирована IT-команда, и она имеет более глубокую экспертизу в нативной разработке, чем в кроссплатформенной, то предпочтение скорее отдаётся нативной разработке.

Но при всём этом стоит понимать, что для того, чтобы разрабатывать продукт под две различные операционные системы, довольно сильно отличающиеся друг от друга, требу-

ется собирать две отдельные команды специалистов. Одна из команд будет писать версию приложения на языке Swift (под iOS), а вторая команда на JAVA или Kotlin (под Android). А это уже будет обходиться компании в двойную стоимость.

Но в рамках данной работы освещаются кроссплатформенные средства разработки. Поэтому на вопрос о выборе подходящего кроссплатформенного инструмента поможет ответить **набор критериев** оценки фреймворков, своего рода **базис**, на который стоит опираться при выборе того или иного средства разработки.

5. Некоторые критерии оценки средств кроссплатформенной разработки

Выбор кроссплатформенного инструмента для проекта в общем может основываться на следующих показателях:

- **Скорость разработки** Под скоростью разработки понимается то, как быстро удастся воплотить идею в готовое работающее приложение (путь от идеи до внедрения). Этому способствует много факторов. Но, например, фреймворки Flutter и React Native обладают передовым свойством — «hot reload», что позволяет отслеживать изменения в режиме реального времени (как только меняется часть кода, изменённый внешний вид приложения можно увидеть в среде разработки). Данная функция, в частности, позволяет достичь относительно быстрого цикла разработки по сравнению с другими фреймворками, такими как Apache Cordova и Angular. Также и готовые компоненты пользовательского интерфейса и библиотеки могут помочь сократить время и затраты, необходимые для создания приложения.

- **Производительность** Это о том, как инструмент проявляет себя с точки зрения скорости работы приложения, стабильности и отзывчивости. Фреймворки Flutter и React Native предназначены для создания высокопроизводительных («быстрых и отзывчивых») приложений, которые могут соответствовать или превосходить по производительности приложения, созданные с помощью нативных инструментов разработки. Apache Cordova и Angular, напротив, не способны предоставить такой же уровень производительности, но для более простых приложений они более чем достаточны.

- **Пользовательский интерфейс** Насколько просто создавать и настраивать пользовательские интерфейсы с помощью конкретного инструмента кроссплатформенной разработки? Поддерживает ли он «родные» компоненты и стилизацию, благодаря которым приложение может выглядеть и ощущаться как «родное» приложение? Предлагается ли гибкая система верстки и поддержка нескольких размеров и разрешений экрана?

- **Пользовательский опыт (user experience—UX)** Здесь речь идёт о внешнем виде и опыте взаимодействия с интерфейсом. Приложение может «ощущаться» как родное, либо быть гибридным или иметь веб-подобный интерфейс. Способен ли инструмент кроссплатформенной разработки обеспечить нативный пользовательский опыт с плавными анимациями и переходами? Поддерживает ли он такие функции, как автономное хранение, push-уведомления и обновления в реальном времени и другие? Flutter и React Native позволяет создавать приложения с нативными компонентами и пользовательским интерфейсом. В свою очередь Apache Cordova и Angular для создания своих интерфейсов используют веб-технологии.

- **Интеграция** В данном случае подразумевается, с какими иными технологиями можно и необходимо будет интегрировать приложение. Некоторые фреймворки могут иметь лучшую совместимость с определенными технологиями, чем другие. Важно понимать, поддерживает ли конкретный инструмент популярные бэкенд-сервисы (например, такие как Firebase или AWS), можно ли подключаться к собственным API и библиотекам сторонних производителей. Поэтому при выборе инструмента важно учитывать потребности в интеграции и каждый случай разбирать отдельно.

- **Опыт команды разработчиков** Немаловажно понимать, какими языками программирования и технологиями владеет команда разработчиков, ведь, как правило, фреймворки предусматривают знания определенных языков программирования. Вообще говоря, изучение (переучивание) нового языка может сказаться на скорости разработки. Необходимо учитывать уровень квалификации команды.

- **Поддержка сообществом** Ключевым фактором является, насколько велико и насколько активно функционирует и развивается сообщество программистов вокруг конкретного фреймворка. Также важно, чтобы у разработчиков был доступ к различ-

ным ресурсам, посвящённым фреймворку, и документации. Некоторые фреймворки имеют более крупные сообщества и более развитые сети поддержки, чем другие. Это напрямую влияет на поиск решений, на будущее конкретного кроссплатформенного инструмента, на постоянную поддержку и обновления.

Оценивая каждый инструмент кроссплатформенной разработки по данным критериям (факторам), можно примерно определить, какой из них лучше всего подходит для конкретных требований.

Но более подробный анализ и аналитику можно провести по следующему набору метрик (критериев).

6. Метрики оценки фреймворков мобильной разработки

6.1. Определения и примечания

- hot-reload — функция горячей перезагрузки, которая позволяет разработчикам видеть результаты изменений кода практически мгновенно, не требуя полной перестройки или перезапуска приложения. Данная функция позволяет сэкономить значительное количество времени на разработку и упростить процесс отладки.
- react-native fast refresh [13] — аналог функции hot-reload фреймворка Flutter
- Skia — это библиотека 2D-графики с открытым исходным кодом, которая предоставляет общие API-интерфейсы, работающие на различных аппаратных и программных платформах (в том числе и Flutter)

6.2. Набор критериев и оценка фреймворков

	ЯП, лежащий в основе	Среда разработки и установка	Поддерживаемые версии мобильных устройств	Наличие функции hot-reload	Наличие внутреннего отладчика	Производи-тельность (в среднем)	Библиотеки/ плагины	Сообщество
Flutter	Dart	* Flutter SDK (Flutter, Dart) * CLI * IDE (например, Android Studio)	Текущая версия: 3.7.6 (02.03.2023) Android: API 16 (Android 4.1) + IOS: iOS 11 +	+	Flutter built-in Debugger	60-120 кадров/сек (120 на устройствах, способных обновлять изображение с частотой 120 Гц)	+	+
React Native	JavaScript	* Node.js * React Native CLI * IDE, поддерживающий JavaScript (например VS Code или WebStorm)	Текущая версия: 0.71 (январь 2023) Android: API 21 (Android 5.0) + IOS: IOS 12 +	+	React Native built-in Debugger (есть возможность отладки как в браузере, так и в IDE)	60 кадров/сек	+	+
Apache Cordova	HTML CSS JavaScript	* Node.js * пакетный менеджер npm * Apache CLI * IDE (например, VS Code или WebStorm)	Текущая версия: 11 (декабрь 2021) Android: API 22 + IOS: IOS 11 +	–	Chrome Remote Debugging, Weinre	60 кадров/сек	+	+
Angular	TypeScript	* Node.js * пакетный менеджер npm * Angular CLI * IDE, поддерживающий TypeScript (например, VS Code или WebStorm)	Текущая версия: v15 (ноябрь 2022) Android: 2 последние версии IOS: 2 последние версии	+ – (Hot Module Replacement, команда "ng serve")	Chrome DevTools (как расширение для браузера)	60 кадров/сек	+	+

Рис. 1: *Набор критериев (часть 1)*

	Стилизация	Интеграция и совместимость	Главный минус	Возможность создать прототип (MVP) приложения в короткие сроки	Высокая производительность и сложная логика
Flutter	Встроенная система стилизации (виджеты), можно использовать пользовательские темы для применения единых стилей во всем приложении	Ограниченная поддержка интеграция с native-кодом и native-API, а также со сторонними библиотеками	Ограниченная поддержка native-API и сторонних библиотек	Высокая	Хорошие инструменты для создания высоко производительных нативных мобильных приложений
React Native	Система стилей, похожая на CSS Стили можно задавать с помощью объектов стилей или внешних таблиц стилей	Есть поддержка интеграции с native-кодом, но возможны проблемы совместимости со сторонними библиотеками	Ограничения при создании сложных элементов пользовательского интерфейса	Высокая	
Apache Cordova	Стандартные веб-технологии, такие как HTML, CSS и JavaScript, для стилизации приложений	Ограниченный доступ к некоторым native-API и низкоуровневым системным функциям	Ограничения в производительности и масштабируемости	Средняя	Подходят для создания мобильных приложений с использованием веб-технологий, или одностраничных приложений, с относительно простой логикой
Angular	Установка стилей в компоненте, или с помощью подключения внешних css-файлов	Есть поддержка сторонник библиотек	Сложнен в освоении, более сложный процесс создания мобильного приложения (в отличии от веб-приложений)	Низкая	

Рис. 2: *Набор критериев (часть 2)*

6.3. Выводы и замечания

- Flutter и React Native отличаются своей высокой производительностью, в то время как Angular и Apache Cordova менее производительны, поскольку разработка на них основана на веб-технологиях.

- Приложения на Apache Cordova и Angular являются гибридными приложениями и используют общий код для разных платформ, а далее этот общий код запускается в WebView на целевой платформе (что и ограничивает производительность).

- Интеграция с backend-сервисами: Flutter и React Native имеют удобную интеграцию с backend-сервисами, посредством Firebase и Node.js, соответственно. Angular и Apache Cordova также имеют интеграцию с backend-сервисами, но она не такая простая в том смысле, что может потребовать дополнительной установки и настройки.

- Flutter имеет собственный высокопроизводительный движок рендеринга, за счёт чего удаётся добиться высокой производительности и более приятной, плавной и уникальной анимации. Но, вообще говоря, оценка уровня графической производительности каждого фреймворка является нетривиальной задачей, поскольку это зависит от множества факторов, включая сложность пользовательского интерфейса, количество анимаций и устройство, используемое для запуска приложения. В целом, каждый из фреймворков способен обеспечить "неплохую"графическую производительность, но только в "идеальных условиях"(то есть при соблюдении ряда факторов). Этого можно добиться с помощью оптимизации, минимизируя сложность пользовательского интерфейса, уменьшая количество анимаций и используя оптимизацию для конкретной платформы.

- Все представленные фреймворки имеют достаточно большое сообщество, которое развивается уже годами. А также для работы с каждым фреймворком существует множество библиотек и плагинов. Но в последние годы сообщество и ресурсы вокруг Flutter и React Native растут гораздо быстрее. Это облегчает поиск решений и поддержку при возникновении проблем. Angular и Apache Cordova имеют менее активное сообщество.

- Фреймворки Flutter и React Native позволяют разработать прототип (или MVP [3]) приложения в кратчайшие сроки. Это достигается в разной степени за счёт функции hot-reload, удобства разработки и большого количества внешних библиотек. Angular и Apache Cordova часто требуют больше времени и усилий для разработки, поскольку данные инструменты основаны на веб-технологиях и могут потребовать немного больше конфигурации и настройки для начала работы. А именно следует учитывать различия в стилях и компонентах в зависимости от целевой платформы; может потребоваться создание новых наборов компонент или настройку стилей. В зависимости от платформы навигация и переход между экранами могут быть устроены по-разному. Поэтому при разработке кроссплатформенного приложения надо учитывать эти различия и настраивать маршрутизацию и навигация, чтобы они работали согласованно на разных платформах. Однако обе технологии предлагают широкий спектр инструментов и библиотек, которые могут помочь ускорить процесс разработки, особенно для веб-приложений.

- Компиляция в нативный код на Flutter и React Native: код компилируется в нативный, который выполняется непосредственно на устройстве пользователя. Это позволяет добиться высокой производительности и лучшего взаимодействия с аппаратными возможностями устройства.

[3]Minimal Viable Product (минимально жизнеспособный продукт) — продукт, обладающий минимальными, но достаточными функциями для удовлетворения первых потребителей. Основная задача — получение обратной связи для формирования гипотез дальнейшего развития продукта

7. Примеры программ

Рассмотрим простейшую программу «hello, world».

Flutter:

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      home: Material(
        color: Color.fromARGB(255, 255, 255, 255),
        child: Center(
          child: Text(
            'Hello World',
            style: TextStyle(fontSize: 32),
          ),
        ),
      ),
    ),
  );
}

```

React Native:

```jsx
import React from 'react';
import {Text, View} from 'react-native';

export default function App() {
  return (
    <View
      style={{
        flex: 1,
        alignItems: 'center',
        justifyContent: 'center'
      }}>
      <Text>Hello, World!</Text>
    </View>
  );
}
```

Angular (на Apache Cordova аналогичный html-файл):

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World App</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>

```

Рис. 3: *Простейший пример программы*

Данные примеры демонстрируют, как каждая из программ отличается по используемому языку программирования и фреймворку. Flutter использует Dart, React Native использует JavaScript, а Apache Cordova и Angular используют, в частности, HTML (а также CSS и JavaScript). Замечание: были опущены остальные папки и файлы, которые отображаются в проекте в среде разработки (например, Android Studio), так как в зависимости от используемого фреймворка они сильно отличаются и по большей части специфичны для конкретного средства разработки и разрабатываемого проекта. Для более сложных проектов набор инструментов необходимых для сборки итогового продукта будет отличаться.

Код каждой программы структурирован по-разному, это отражает различия в способах работы каждого фреймворка. Однако на высоком уровне каждая программа просто отображает текст "Hello, World!"по-разному. Но при разборе более сложного приложения разница будет становиться всё заметнее (в силу разных используемых библиотек, синтаксиса, семантик).

Но если обратиться к таким критериям, как «Опыт команды разработчиков» и «Поддержка конкретного фреймворка сообществом разработчиков», то можно сделать следующее заключение: очень удобно сделать выбор в пользу того или иного средства разработки, если уже приходилось работать с ним или знакомы базовые принципы (например, язык Dart довольно-таки похож на Python и C++ в плане синтаксиса и классовой модели, а данные языки, например, изучаются на многих специальностях в технических ВУЗах, гораздо реже студентам приходится иметь дело с JS, HTML, CSS [данное замечание сделано на основе программы и изучаемых языков программирования на факультете ВМК МГУ]).

8. Инфографика по использованию фреймворков

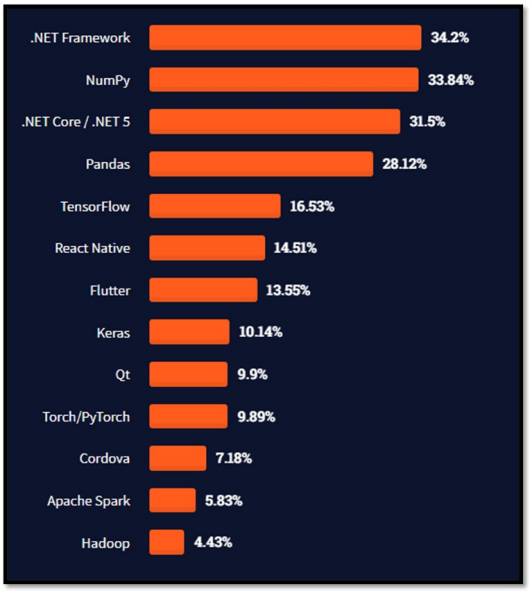

Рис. 4: *[7] Опрос Stack Overflow разработчиков об используемых ими технологиях (2021 год)*

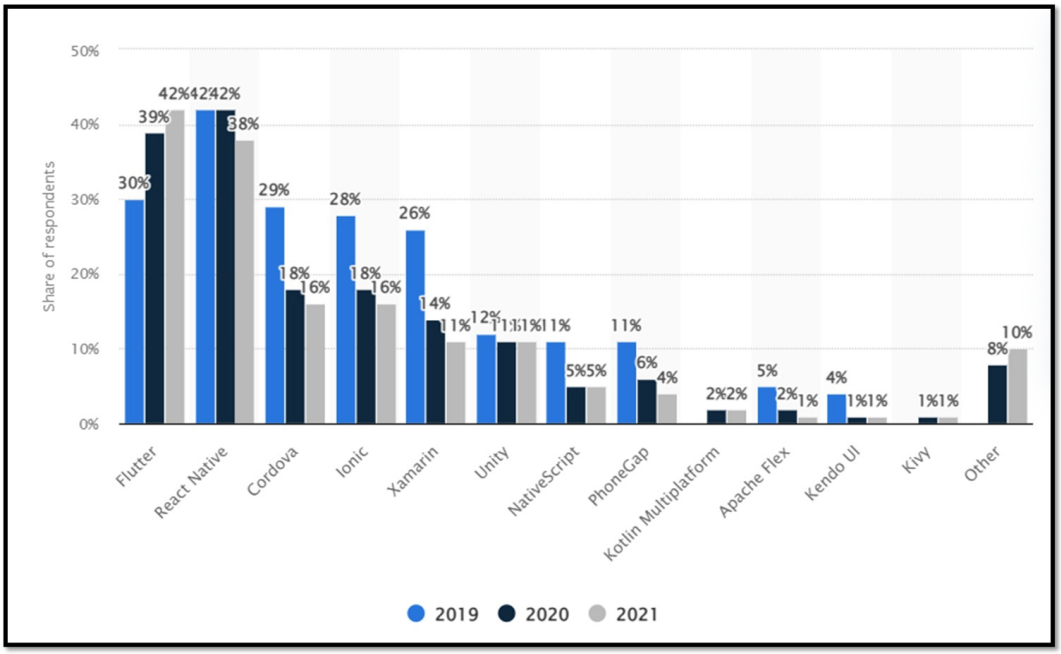

Рис. 5: *[9] Популярность кроссплатформенных фреймворков (2019–2021), источник: Statista.com*

9. Заключение

Каждый из представленных четырёх фреймворков имеет свои уникальные сильные и слабые стороны. Поэтому важно понимать, какой функциональности можно добиться, используя ту или иную платформу при разработке.

Выбор фреймворка для мобильной разработки зависит от многих факторов: требований проекта, технического задания, опыта разработчика, целевой аудитории (конечного пользователя).

Как итог, в ходе данной работы был разработан и подготовлен набор критериев (метрик оценки), по которым может производиться анализ и выбор фреймворка для разработки программного продукта. В рамках выбранного набора метрик были описаны следующие фреймворки для кроссплатформенной разработки: Flutter, React Native, Angular, Apache Cordova.

Список литературы

[1] Flutter web site, info and documentation: *www.flutter.dev*

[2] Dart web site, info and documentation: *www.dart.dev*

[3] React Native web site, info and documentation: *www.reactnative.dev*

[4] Apache cordova web site, info and documentation: *www.cordova.apache.org*

[5] Angular web site, info and documentation: *www.angular.io*

[6] Хашаев, А. *Лекции «Архитектура и процесс разработки ПО»*

[7] Stack Overflow web site: *www.insights.stackoverflow.com/survey/2020 (2021)*

[8] GitHub web site: *www.github.com*

[9] About worldwide-software-developer-working-hours: *www.statista.com*

[10] Flutter vs React Native *www.backendless.com*

[11] How to choose the right cross platform tool www.kotlinlang.org

[12] About sensors in angular link

[13] React native hot-reload functionality link

[14] Henry Andersson, A Comparison of the Performance of an Android Application Developed in Native and Cross-Platform: Using the Native Android SDK and Flutter link

[15] Daniele Palumbo, The Flutter Framework: Analysis in a Mobile Enterprise Environment link