СРАВНЕНИЕ НАТИВНОЙ И КРОССПЛАТФОРМЕННОЙ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Миненков Г.А.

В работе рассматриваются и анализируются основные особенности разработки нативных и кроссплатформенных мобильных приложений. Приведены основные достоинства и недостатки двух подходов.

Базовым подходом к разработке мобильных приложений является нативная разработка. При нативной разработке используются оригинальные языки программирования и инструменты мобильной операционной системы. Разработка iOS приложений ведется в интегрированной в OS X и iOS среде Xcode, на языках Objective-C, Swift, C и C++. Для разработки Android приложений используется среда Android Studio и язык Java. Среда разработки – это специальный комплекс инструментов, созданный для максимально удобного написания кода, проектирования интерфейса, отладки, мониторинга и сборки приложения [1]. Преимуществами нативной разработки являются:

1. Скорость работы приложения. Скомпилированный код проекта оптимален для родной платформы. Приложение получает полную аппаратную поддержку устройства и использует многопоточность для сложных задач. В процессе разработки приложения программисты могут измерять скорость работы всех участков кода и при необходимости их оптимизировать.
2. Гибкость в реализации. Нативная разработка использует все возможности мобильной операционной системы.
3. Новая программная и аппаратная функциональность доступна для реализации сразу после выпуска обновлений.
4. Простота тестирования. В распоряжении разработчиков и тестировщиков есть целый комплекс технологий. Все параметры системы в процессе работы приложения контролируются автоматически, например, если приложение стало использовать больше памяти или ресурсов процессора, это можно легко обнаружить с помощью средств мониторинга. В нативной разработке доступны широкие возможности автоматического тестирования для любого метода в приложении, например, если часть приложения перестанет корректно работать после изменений кода, в таком случае новая версия не соберется, а программист сразу станет ясна причина проблемы. Для нативных проектов встроена функциональность удаленного мониторинга ошибок, которая позволяет увидеть ошибку и ее причину на устройстве пользователя.
5. Поддержка приложений со стороны App Store и Google Play. Apple и Google предъявляют высокие требования к качеству приложений в магазинах. Компании заинтересованы в том, чтобы пользователи их операционных систем получали максимально положительный опыт использования, поэтому каждая часть приложения обязана соответствовать высоким стандартам качества. Например, если изображения используемые при разработке приложения имеют недостаточно высокое разрешение, модераторы App Store откажут в публикации программного средства.

Среди недостатков нативной разработки можно выделить следующее:

1. Стоимость разработки. Чаще всего заказчику необходимо воспользоваться услугами сразу двух команд разработки, что приводит к соизмеримому увеличению затрат.
2. Высокая вероятность различий в работе приложений, ввиду отсутствия единой кодовой базы бизнес-логики.

Альтернативным и довольно распространенным решением на сегодняшний день является кроссплатформенная разработка. Для кроссплатформенной разработки используются специальные инструменты (Unity, Xamarin, ReactNative), которые позволяют создавать приложения сразу для нескольких мобильных операционных систем. Преимущества кроссплатформенной разработки:

1. Экономия бюджета. Использование одной технологии и набора графики снижает количество рабочих часов и бюджет проекта.
2. Время разработки. Отсутствие уникальных элементов интерфейса и одна технологическая платформа сокращает сроки разработки.
3. Поддержка и обновление продукта. Добавление функциональности или исправление ошибок применяется сразу для всех платформ.
4. Мобильная версия сайта. Большинство кроссплатформенных решений позволяют генерировать мобильную версию сайта из приложения.
5. Единая логика приложения. Логика приложения будет одинаково работать для всех платформ. Написанная и отлаженная логика содержит потенциально меньшее количество ошибок и расхождений в своей работе.

Проблема существования двух разных подходов в области разработки интерфейсов мобильных приложений существует довольно долго. Первым значимым отличием в опыте использования устройств на базе iOS и Android является наличие навигационных клавиш для телефонов на базе Android и их отсутствие на iOS устройствах. На рисунке 1 изображена навигационная панель Android устройства.

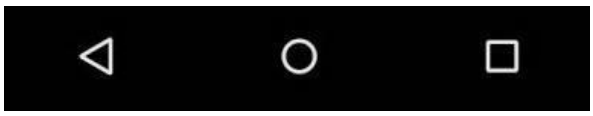

Рисунок 1 – навигационная панель Android

Наличие навигационной панели в Android и её отсутствие в iOS диктует определенные практики в реализации навигации по приложению. Причиной подобных решений является то, что основным способом навигации назад в iOS приложениях является движение пальцем слева направо, в Android приложениях подобную функцию выполняет кнопка назад, изображенная слева на рисунке 1.

Однако, несмотря на значительные отличия некоторых компонентов приложения, на данный момент довольно сильно распространены кроссплатформенные решения с единым интерфейсом для двух операционных систем. Ярким примером такого приложения является популярная социальная сеть Instagram. Пример единого интерфейса Instagram для iOS и Android изображен на рисунке 2. Единственным заметным отличием для двух приложений являются иконки.

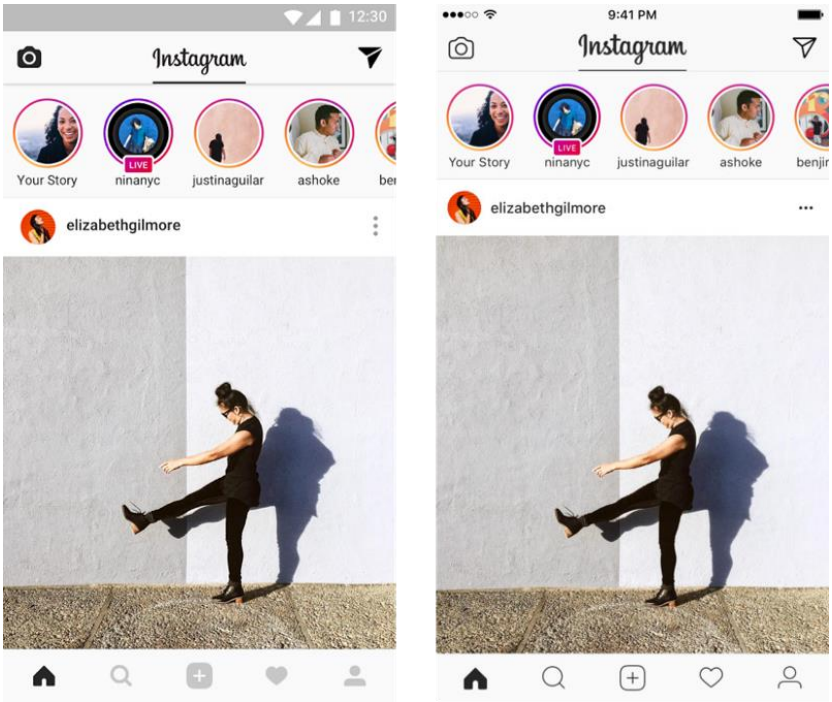

Рисунок 2 – интерфейс приложения Instagram для Android и iOS

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

Список использованных источников:

1. WINFOX [Электронный ресурс]. – Электронные данные. – Режим доступа: http://wnfx.ru/nativnaya-ili-krossplatformennaya-razrabotka-chto-luchshe/. – Дата доступа: 12.04.2019.