УДК 004

СРАВНЕНИЕ НАТИВНОЙ И КРОССПЛАТФОРМЕННОЙ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

[1]Крук Д. А., [2]Киселевич О. А., [3]Михейчик А. А.
Белорусский национальный технический университет,
Минск, Беларусь, kruk@mail.ru,
Белорусский национальный технический университет,
Минск, Беларусь, kiselevich@mail.ru,
Белорусский национальный технический университет,
Минск, Беларусь, miheichik@mail.ru,
Научный руководитель – ст. преподаватель, Петровская Т. А.

Аннотация. Разработка приложений, плюсы и минусы нативной и кроссплатформенной разработки.

Введение. Мобильная разработка делится на нативную икроссплатформенную.

Нативная разработка подразумевает под собой создание отдельного приложение под каждую платформу (IOS, Android). Основными языками для iOS разработки является Objective-C (1983 год, Бред Кокс), Swift (2014 год. Apple), для Android – Java (1995 год, Джеймс Гослинг), Kotlin (2011 год, JetBrains).

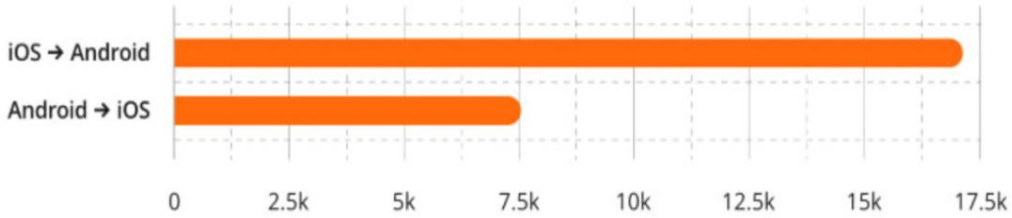

Рисунок 1 – График перехода с IOSна Android в обе стороны

Из данного рисунка, мы видим, что чаще всего, компании, имеющие приложение на iOS создают и на Android, реже компании добавляют приложение на систему iOS.

Кросс платформенная разработка предполагает, разработку одно приложение для обеих платформ. Используется ReactNative (2015 год, Facebook), Flutter (2017 год, Google).

Основная часть. Рассмотрим плюсы и минусы моделей разработки.

В нативной разработке имеются следующие плюсы:

1. Высокая скорость и производительность, связанная с тем, что у языков, используемых в нативной разработке, имеется больший доступ непосредственно к платформе, что позволяет увеличить скорость отклика, а также более плавный переход.

2. Больше возможностей для интеграции с платформой – благодаря возможности работать на прямую с платформой, что позволяет увеличить доступ к

ее функциям, создаваемых приложения лучше работают, где имеется связь с фотографией, Bluetooth, NFC, GPS, аудиомодулю и другим функциям.

3. Привычный пользовательский интерфейс (более адаптированный) – из-за высокой интеграции с платформой, проще установить привычный дизайн.

4. Лучшее позиционирование в онлайн-маркетплейсах–проще выпускать на площадках, так как имеется меньше требований и не такая длительность проверки приложения, а также, в теории, на маркетплейсах данных платформ есть ранжирование, которое поднимает «родные» приложения.

Минусы нативной разработки:

1. Время и стоимость создания приложения – из-за того, что нужно писать два различных приложения под каждую платформу увеличивается количество нужных разработчиков и времени, что в свою очередь увеличивает цену на разработку.

2. Зависимость от одной мобильной операционной системы

3. Упущенные финансовые возможности (прибыли) – имея приложение только на одной платформе, автоматически исключается прибыль от людей, имеющих другую платформу [1].

В кроссплатформенной разработке имеются следующие плюсы:

– единая база кода для всех платформ.
– сокращение времени и стоимости разработки.
– покрытие более широкой аудитории пользователей.
– допустимость одинакового интерфейса и UX.

Минусы кроссплатформенной разработки:

– меньшая гибкость, чем у нативных приложений.
– более низкая производительность и надежность.
– возможное несоответствие в дизайне на двух платформах.
– возможные сложности с маркетплейсами приложений.

	Средняя ставка (USD)	Количество человеко-месяцев	Количество часов	Итого, USD
iOS/Android	25	2	320	8000
Кроссплатформенные приложения	25	1,5	240	6000

Рисунок 2 – Сравнение затрат

Из рис. 2, видно, что нативная разработка требует больших затрат как человеческого ресурса, так и денежного.

По статистике кросс платформенных приложений на маркетплейсах:

– Flutter – это 0,4 % App Store и 3,4 % Google Play. Самые большие – SHEIN, Alibaba.com и WeChat.

– React Native – 2,6 % App Store и 4,4 % Google Play. Самые большие – TikTok и Instagram[2].

Заключение. Выбор модели разработки, а также самого языка, зависит от клиента, а также от приложения которое он хочет создать, на данный момент

Flutter, хоть и набирает рост, однако является молодым языком, в котором еще не все процессы отлажены, благодаря частным обновлениям данного языка, возможно, что вскоре он сможет заменить нативную разработку, а также Flutter на данный момент уже используется в веб- разработке, но нативные приложения, все еще актуальны, и порой, лучше выбрать именно их, не смотря на стоимость разработки.

Литература

1. Разработка мобильных сервисов и приложений [Электронный документ]. – Режим доступа: https://appcraft.pro/blog/razrabotka-prilozhenij-dlja-android/. – Дата доступа: 01.11.2022.

2. Этапы разработки мобильного приложения: аналитика и техническое задание [Электронный документ]. – Режим доступа: https://vc.ru/dev/142571-etapy-razrabotki-mobilnogo-prilozheniya-analitika-i-tehnicheskoe-zadanie. – Дата доступа: 01.11.2022.