На чию користь віддати свій вибір, залежить від ряду критеріїв, від оцінки факторів, які є критичними для певної системи і того, як вони впливають на результати. Мартін Фаулер, автор ряду книг і статей про архітектуру ПЗ, виділяє критерії відштовхуючись від яких можна обрати більш кращу модель для певної системи. Ці критерії наведені у таблиці 2.

На основі викладеного можна зробити такі висновки, що мікросервіси більш трудомісткі, і це окупається тільки для більш складних систем. Тому, якщо можна впоратися зі складністю системи за допомогою монолітної архітектури, не слід піддаватися впливу моди і переходити на мікросервіси. Тому багато провідних фахівців радять починати з монолітної системи і переходити на мікросервіси тільки в тому випадку, якщо ви абсолютно впевнені, що без поділу на сервіси проблему ніяк не вирішити. Наприклад, якщо вартість додавання нового функціоналу починає перевершувати вигоду від цього самого функціоналлу, то слід задуматися про складність рефакторінга великого моноліту і почати планувати перехід до мікросервісам.

Список використаних джерел:

1. Никита Цуканов. Архитектура микросервисов: [Электронный ресурс]: URL: http://itnan.ru/post.php?c=1&p=320962

2. Partitioning Problems in Parallel, Pipelined, and Distributed Computing / Bokhari_S. // IEEE Transactions Computers. – 1988 – 57 с.

3. Алексей А.О. Переход от монолита к микросервисам [Электронный ресурс]: URL: https://habrahabr.ru/post/305826/.

4. Никита Цуканов. Проектирование и микросервисы для самых маленьких [Электронный ресурс]: URL: https://habrahabr.ru/post/311208/

5. Ньюмен С. Создание микросервисов. – СПб.: Питер, 2016. – 304 с.: ил. – (Серия «Бестселлеры O'Reilly»).

Обловацька М.В.

студент,

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

ПОРІВНЯННЯ НАТИВНОЇ ТА КРОСПЛАТФОРМНОЇ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

Нативна (від англ. native – рідний, природний) розробка передбачає використання оригінальних мов та інструментів розробки тієї чи іншої операційної системи. Наприклад, додатки для Android створюються в середовищі Android Studio за допомогою мови Java, а для iOS – у середовищі XCode за допомогою мов Objective-C, Swift, C та C++ відповідно. Тобто нативними додатками можна назвати ті, з якими користувач зустрічається з першого дня використання пристрою. Такі застосування можуть отримувати доступ до усіх служб та сервісів телефону.

Кросплатформна розробка передбачає використання різних фреймворків для створення додатків на мові JavaScript. Структура та логіка додатку створюється за допомогою таких інструментів як Appcelerator Titanium, Adobe PhoneGap, Xamarin та ін. мовою JavaScript, а потім інтегрується в базовий проект для Android Studio або XCode, тобто дозволяє створювати проект однією і тією ж самою логікою для різних операційних систем. Можна сказати, що кросплатформні додатки це щось на зразок мобільних сайтів, яким не завжди потрібен інтернет, а з точки зору дизайну вони ближче до мобільних застосувань.

Існує ряд відмінностей між нативною та кросплатформною розробками мобільних додатків, тому обирати якому саме способу розробки надати перевагу слід виходячи з функцій та цілей проекту. Нижче наведено переваги обох методів розробки.

Нативна розробка має наступні позитивні риси:

1. Швидкість роботи додатку. Коли проект вимагає обробки великих обсягів даних, інтенсивність використання пам'яті пристрою значно зростає. Якщо додаток створюється за допомогою оригінальних інструментів розробки, то отриманий в результаті компіляції проекту код буде оптимальним для даної платформи. Додаток отримуватиме апаратну підтримку пристрою, буде використовуватися багатопоточність для реалізації складних задач і завантаження контенту в фоні.

2. Підтримка Google і Apple. Обидві компанії зацікавлені в тому, щоб користувач отримав позитивний досвід при використанні додатку на відповідній платформі. Тобто додаток має виглядати якісно і працювати швидко (чого досить важко досягти кросплатформними технологіями), інакше – його просто не пропустять в магазин (Google Play або App Store).

3. Використання останніх технологій. Новий програмний та апаратний функціонал, наданий компаніями-виробниками пристроїв та операційних систем, стає доступний для реалізації відразу після випуску відповідних оновлень.

4. Гнучкість в реалізації. При використанні нативної розробки реалізувати можна все, на що здатні технології тієї чи іншої мобільної операційної системи, на відміну від обмежень в побудові інтерфейсу і складності візуальних ефектів, що накладаються фреймворками для кросплатформної збірки проектів.

5. Досвід користувачів. Найперше чого чекає користувач від додатку – швидкого відгуку на свої дії. Тобто прокрутка сторінки та анімація мають бути плавними, без підвисань. Кросплатформні додатки в цьому плані значно поступаються нативним. Також користувач очікуватиме, що кожен елемент управління на екрані матиме стандартний вигляд, а для різних платформ ці стандарти є різними.

6. Безпека. Якщо у додатку потрібен особливий рівень шифрування, то забезпечити надійний рівень захисту даних можливо тільки при нативній розробці, адже це зв'язано з математикою, а подібні операції вимагають максимально ефективного використання апаратних ресурсів.

Кросплатформний підхід до розробки має такі переваги:

1. Використання єдиної логіки додатку для усіх платформ. Досить часто це може бути і мінусом через різну архітектуру операційних систем, однак написана і налагоджена один раз логіка містить потенційно меншу кількість помилок і розбіжностей в своїй роботі: розробнику не доведеться проробляти подвійну роботу з пошуку проблем на кожній платформі.

2. Час розробки. Відсутність унікальних елементів інтерфейсу і одна технологічна платформа скорочує терміни розробки.

3. Мінімізація витрат. Потрібно менше ресурсів для реалізації програми відразу під кілька платформ. В цьому, власне, і суть кросплатформного підходу – один і той же код працює і на Android, і на iOS. Розробників, які займаються проектом, потрібно рівно в два рази менше. Дизайнер робить тільки один набір графіки. Все це знижує кількість робочих годин і, як наслідок, бюджет проекту.

4. Максимізація прибутку. Кросплатформні додатки можуть збільшити прибуток від реалізації проекту шляхом впливу на більше число користувачів, адже компанія буде мати додаток в більшості середовищ.

5. Можливість централізувати усі зусилля команди на запуску одного вихідного коду. Це спрощує цикл оновлення продукту. Якщо в проект потрібно щось додати або виправити якусь помилку, то це робиться відразу для усіх платформ. Фрагментація при реалізації MVP та оновленні відсутня.

6. Немає необхідності включати нових розробників у команду. Це пояснюється можливістю використання мобільної версії сайту. Більшість кроссплатформних рішень використовують сімейство мов JavaScript, тому якщо у проекту вже є мобільна версія сайту, значна частина коду і матеріалів може бути використана в додатку без змін. Тобто достатньо мати команду розробників, котрі розробляли повну та мобільну версії сайтів і мають знання HTML CSS JavaScript.

Отже, порівнюючи нативну та кросплатформну розробки, можна зробити висновок про те, що з технічної точки зору, нативна розробка має набагато більше плюсів. Однак, до вибору тієї чи іншої стратегії завжди приводять індивідуальні обставини. Існують сфери, в яких використання кросплатформних технологій є виправданим: це ігровий сектор і тестові проекти. Для не ігрових проектів, спрямованих на довгостроковий розвиток, нативна розробка залишається єдиним варіантом.

Список використаних джерел:

1. Scott Olson, John Hunter, Ben Horgen, Kenny Goers. Professional Cross-Platform Mobile Development in C# // Indianapolis, 2012. – 357 с.

2. Joe Conway, Aaron Hillegass. iOS Programming The Big Nord Ranch Guide // Atlanta, 2012. – 589 с.

3. Bill Phillips, Brian Hardy. Android Programming: The Big Nerd Ranch Guide // Atlants, 2013. – 733 с.