История развития мобильных устройств.

Появление первого мобильного телефона.

В различных операционных системах с каждым днем растет количество пользователей мобильных телефонов, которые понимают, что с помощью своего телефона они получают доступ к большому количеству информации т.е. имеют возможность просматривать медиа-контент, устанавливать полезные программы и игры, а также планировать свой отпуск. Благодаря этому рынок мобильных приложений является перспективной сферой, в которой задействовано большое количество людей. С помощью мобильных приложений специалисты по связям с общественностью могут поддерживать связь с потребителями из любого места, что позволяет налаживать тесные контакты со своей целевой аудиторией.

Современный человек проводит с мобильным телефоном практически все свободное время. Когда он видит сообщение от той или иной программы, то может зайти в нее и ознакомится с контентом сообщения. Таким образом, разработчики мобильных приложений ведут конкурентную борьбу за внимание своей целевой аудитории, используя различные маркетинговые методы. При помощи цифровых технологий можно не только выпускать развлекательные программы или игры но и также предлагать клиентам свои продукты и услуги, собирать с них отзывы и т.д.

"Мобильное приложение" - это компьютерная программа, созданная специально для использования в мобильном телефоне, смартфоне или коммуникаторе, которая предназначена для выполнения той или иной задачи. На первых мобильных устройствах было малое количество мобильных приложений, поставляющихся совместно с программным обеспечением устройства. Первым мобильным приложением можно считать телефонный справочник - та часть программного

обеспечения, которая упорядочивала контакты".

Motorolla DynaTac 8000X в 1983 году стал мобильным телефоном, с которого все началось и который проложил путь для всех других мобильных телефонов, включая смартфоны и приложения, без которых современные пользователи не могут жить. В то время как гиганский кирпичик телефона мог делать немного больше, чем звонить, Nokia и другие производители решили расширить функции, добавив простые игры, такие как Snake, Pong, Tic-Tac-Toe.

\*\*WAP и первое подключение к интернету\*\*<br>

WAP (Wireless Application Protocol) появился в 1998 году, и именно он объединил интернет и мобильную связь. Теперь можно было встроить в телефон браузер, установить соединение с серверами и получить данные на устройство.

![wap](./Wireless+Application+Protocol+(WAP).jpg)

Одним из первых телефоном с WAP-браузером был Nokia 7100, который выпустили в 1999 году. Тогда же начали появляться компании, разрабатывающие продукты специально для мобильных устройств.

WAP дал людям не только игры, но и возможность читать новости с мобильного, пользоваться электронной почтой, загружать карты и даже бронировать билеты. Для этого создавались WAP-сайты со специальной разметкой для мобильных экранов. Это были простые страницы из текста и ссылок, почти без картинок.

WAP и первое подключение к интернету

WAP (Wireless Application Protocol) появился в 1998 году, и именно он объединил интернет и мобильную связь. Теперь можно было встроить в телефон браузер, установить соединение с серверами и получить данные на устройство.

Одним из первых телефоном с WAP-браузером был Nokia 7100, который выпустили в 1999 году. Тогда же начали появляться компании, разрабатывающие продукты специально для мобильных устройств.

WAP дал людям не только игры, но и возможность читать новости с мобильного, пользоваться электронной почтой, загружать карты и даже бронировать билеты. Для этого создавались WAP-сайты со специальной разметкой для мобильных экранов. Это были простые страницы из текста и ссылок, почти без картинок.

Эволюция в вашем кармане: как развивались мобильные приложения

WAP-версия сайта Билайна

В то время компании еще не оценили телефоны как способ коммуникации с клиентами, поэтому таких решений было катастрофически мало.

Например, Ericsson сделали путеводитель Michelin: через WAP была доступна база из 60 000 отелей и ресторанов Европы. Еще несколько примеров использования WAP для бизнеса приведены в White Paper Nokia 1999 года:

Клиенты Deutsche Bank и Visa International могли получать информацию о последних транзакциях, просматривать баланс и оплачивать счета;

Пассажиры авиакомпании Finnair могли бронировать билеты и получать информацию о рейсах.

Первая открытая ОС для разработчиков.

В 2001 Symbian стала открытой операционной системой, и в это же время появилась Nokia 7650, на которую можно было устанавливать приложения от сторонних разработчиков. Это должно было стать прорывом на рынке, но бума не случилось из-за сложностей разработки и ограниченных возможностей смартфонов.

![symbyan](./1-1.jpg)<br>

У разработчиков был бедный выбор средств разработки для Symbian. Основной язык С++ был сложным в изучении и компиляции. Приходилось изворачиваться, чтобы написать приложение, которое было бы совместимо с большинством устройств на Symbian. Также многих отпугивала необходимость покупки сертификатов безопасности для подписи приложений.

В это же время развивался рынок Java-приложений. Разработка приложения на Java занимала меньше времени и подходила для Windows Mobilе, Android, bada, Palm OS и BlackBerry OS. В Symbian также поддерживалось подмножество Java — J2ME, но функциональность таких приложений была сильно ограничена, поэтому разработкой на Java под Symbian практически никто не занимался.

В Nokia никак не стремились помогать разработчикам развивать рынок. Все было настолько плохо, что в 2005 году вышла Symbian 9.1, которая была не совместима с приложениями, выпущенными для предыдущих версий. Каждое приложение требовало доработки.

У разработчиков не было нормальной среды для создания проектов, большинство использовали Eclipse, предназначенной изначально для разработки на Java. Nokia выпустили инструмент для разработки на C++ — Carbide на основе Eclipse, но большая часть его возможностей была платной. Лицензия стоила от 300 до 8000 евро, это сильно влияло на конечную стоимость приложения.

При этом развитием самой Symbian никто особо не занимался, больше внимания уделяли новым дизайнам смартфонов Nokia. На рынок выходили новые модели, а саму ОС обновляли примерно раз в год.

Попытки что-то исправить в Nokia начали предпринимать только в 2009 году. Они пытались решить проблемы с недружелюбным API, дать больше возможностей для создания приложений и упростить разработку с помощью фреймворка Qt, затем открыли исходный код Symbian и объявили о создании Symbian Foundation, что должно было помочь популяризировать ОС.

Но это все не помогло собрать вокруг Symbian сообщество разработчиков и партнеров-производителей смартфонов. В итоге ОС не смогла конкурировать с iOS и Android.

Выход iPhone и запуск App Store.

В 2007 году Стив Джобс представил миру первый iPhone. Тогда много говорили о возможностях нового смартфона, но никто не говорил, как этих возможностей удалось достичь.

![firstIphone](./evolucion-iphone-1-2007.jpg)

Архитектура iOS была похожа на MacOS, но система была полностью закрытой. Джобс не хотел, чтобы сторонние разработчики могли разрабатывать приложения для iOS, и не собирался открывать SDK. Вместо этого он хотел, чтобы разработчики создавали веб-приложения, и дал возможность создавать браузерные закладки на домашнем экране. Мы знаем это из биографии Джобса, которую написал Уолтер Айзексон.

Полноценный движок Safari уже присутствует внутри iPhone. То есть вы можете создавать изумительные Web 2.0 и Ajax приложения, которые выглядят и ведут себя так же, как родные программы iPhone. И они способны прекрасно взаимодействовать с его сервисами: звонить, отправлять электронные письма, разыскивать местоположение в Google Maps. И знаете что? Для этого не нужен SDK!" — Стив Джобс.

Но пытливые умы взломали файловую систему, начали писать инсталляторы для нативных приложений и заодно — сами приложения. Так появился джейлбрейк.

Позже совет директоров Apple все же убедил Джобса легализовать сторонние приложения. В итоге в марте 2008 года iPhone SDK стал доступен всем желающим, а в июле презентовали App Store. Это означало, что Apple берет на себя дистрибуцию продуктов разработки пользователям.

App Store стал толчком к развитию индустрии разработки приложений, но проблемой был Objective-C. Язык программирования для iOS кардинально отличался от популярных тогда скриптовых JavaScript и Flash Action Script. Мало кто хотел тратить время на изучение нового синтаксиса, ведь устройства на iOS занимали еще очень маленькую долю рынка, а основная его часть принадлежала смартфонам на Symbian.

На тот момент двигателем прогресса были игры, а разработка приложений — лишь развлечением для специалистов в смежных областях. На продаже игр уже можно было зарабатывать, а спроса на разработку сервисов и приложений для бизнеса еще не было. Компании не были заинтересованы в разработке приложений для клиентов, потому что:

- Бизнес не доверял мобильным технологиям и никто не знал, как их использовать на благо компаний;

- Существовали ограничения смартфонов в отображении, передачи и приеме данных;

- Приложения не подходили для серьезных задач из-за проблем с безопасностью данных.

\*\*Проблемы кроссплатформенности: когда всем надо все и сразу\*\*<br>

В 2008 появился Android Market, а в 2010 — Windows Mobile Store.

![andriodwinda](./Снимок%20экрана%202023-12-20%20в%2022.30.21.png)

К тому времени мобильный интернет стал доступнее, а в SDK для разных платформ появились решения по безопасности и интеграции. Это был новый этап развития в разработке приложений: от развлечения для народа разработчики перешли к решению задач бизнеса. Но впереди уже маячила другая проблема.

Быстрое параллельное развитие iOS и Android создало двухполярную систему, и разработчикам нужно было поддерживать несколько платформ одновременно. Из кроссплатформенных инструментов были только Flash и обычный мобильный браузер. И то в 2010 году Apple отказались от поддержки технологии Adobe Flash в iOS.

Разработка одного даже самого простого решения под разные платформы уперлась в человеческие, временные и материальные ресурсы. Хотя браузерные технологии были развиты достаточно хорошо, мобильную веб-разработку тормозила низкая производительность смартфонов.

Как все решилось.

На помощь пришли библиотеки компонентов и фреймворки для создания приложений на Android и iOS на базе браузерных технологий без использования языков программирования: Xamarin, Cordova, Phonegap.

С их помощью создается подобие мобильного сайта, сверху накладывается платформенный код, который транслирует вызовы от системы к приложению и обратно.

Эти инструменты решили проблемы маленьких приложений с небольшим функционалом и дали возможность бизнесу быстро и за относительно небольшие деньги протестировать свое присутствие на мобильных платформах. Дальше нужно было смотреть в сторону нативной разработки, потому что остро стояли проблемы производительности, потребления ресурсов, отзывчивости кроссплатформенных приложений и «чужеродного» дизайна.

Развитие кроссплатформенных решений: React Native

В 2015 году разработчики Facebook на конференции React.js Conf представили свой инструмент для кроссплатформенных решений — фреймворк React Native. В нем компоненты приложения, написанные на JS, транслируются в нативные Android и iOS. Этот инструмент принципиально отличается от других систем для создания кроссплатформенных приложений:

- Отсутствием WebView и HTML-технологий;

- Отрисовкой интерфейса. В RN её выполняет ОС устройства, а не браузер;

- Отсутствием дополнительной «обертки» кода — вместо нее JS взаимодействует с ОС через специальный мост. Так в приложении используются нативные компоненты пользовательского интерфейса.

Благодаря этим отличиям приложения, сделанные на React Native, максимально похожи на нативную разработку, и у них меньше проблем с производительностью.

Хотя стало проще и лучше, проблемы все равно остались:

- Знаний одного JS все равно не хватит, потому что работу с возможностями мобильной платформы нужно реализовывать через модули на нативных языках;

- Facebook постоянно что-то переписывает в архитектуре RN, поэтому вечно что-то меняется. Новые релизы часто сопровождаются обратной несовместимостью в коде;

- Скорость работы приложений выше, но все еще не на уровне нативных приложений;

- Нативная разработка становится проще.

Производители стараются минимизировать сложности в разработке и пытаются упростить языки разработки. В результате появились более простые Swift и Kotlin.

Swift представили на конференции WWDC в 2014 году. В нем осталось много от Objective-C, но он работает по аналогии со скриптовыми языками. Код определяется типами переменных, а не указателями. Это делает его изучение легче для тех, кто уже владеет каким-либо скриптовым языком.

Kotlin с 2010 года разрабатывала компания JetBrains. Целью было сделать более лаконичный и простой язык, чем Java, в котором уже накопился багаж неудачных решений. С 2017 язык официально рекомендуется для Android-приложений.

Теперь для разработки нативных приложений есть два языка с простым и гибким синтаксисом, при этом они похожи между собой.

![webpic](./4-3.webp)

Получилось упрощение не только от Objective-C к Swift и от Java к Kotlin, но и от Swift к Kotlin и наоборот. Это гораздо больший шаг к кроссплатформенности, чем разработка приложений на web-view.

История Objective-C и Swift(SwiftUI).

Objective-C был создан в 1980-х годах Брэдом Коксом и Томом Лавком в рамках проекта StepStone. Целью создания Objective-C было сочетание объектно-ориентированного программирования с возможностями языка C.

Первая версия Objective-C, называемая Objective-C 1.0, была выпущена в 1983 году. В 1988 году была выпущена версия Objective-C 2.0, в которой были добавлены новые возможности, включая поддержку исключений и категорий.

Objective-C начал использоваться среди разработчиков Macintosh в конце 1980-х и начале 1990-х годов, когда компания NeXT, основанная Стивом Джобсом после его ухода из Apple, выбрала Objective-C в качестве основного языка программирования для своей операционной системы NeXTSTEP.

В 1997 году Apple приобрела NeXT и начала использовать NeXTSTEP как основу для своей новой операционной системы Mac OS X. Objective-C был выбран в качестве основного языка программирования для разработки приложений для этой операционной системы.

Objective-C быстро стал популярным среди разработчиков iOS и macOS, так как это основной язык программирования для разработки приложений для этих платформ.

Objective-C все еще используется для поддержки существующих приложений, но Swift стал основным языком для разработки приложений под iOS и macOS.

Язык Swift был разработан компанией Apple и впервые представлен в 2014 году. Он является преемником языка Objective-C, который был основным языком программирования для iOS и macOS-приложений.

Основные причины разработки Swift из Objective-C заключались в следующем:

1. Безопасность: Swift предлагает множество возможностей для обработки ошибок и предотвращения возникновения ошибок во время выполнения, что делает код на Swift более безопасным и надежным.

2. Простота использования: Swift имеет более простой и понятный синтаксис, который упрощает разработку приложений и снижает количество возможных ошибок.

3. Высокая производительность: Swift более эффективно работает при выполнении некоторых операций, таких как сортировка и циклы, что приводит к увеличению производительности приложений.

4. Интероперабельность с Objective-C: Swift был разработан с учетом существующего кода на Objective-C, что означает, что разработчики могут использовать уже существующие библиотеки и фреймворки на Objective-C в коде на Swift.

5. Поддержка сообщества: Swift имеет активное сообщество разработчиков, которое постоянно развивает и улучшает язык, предлагая новые возможности и рекомендации по его использованию.

Разница StoryBoard, Swift и SwiftUI.

Storyboard, Swift и SwiftUI - это три разных инструмента, используемых для разработки приложений для iOS и Mac.

Storyboard - это графический редактор интерфейса, который позволяет визуально создавать пользовательские интерфейсы приложения. Он основан на языке Objective-C и использует сторонний формат файла XML, который описывает разметку интерфейса. С помощью Storyboard можно создавать и настраивать экраны, переходы между ними, связи с контроллерами и управлять отображением элементов интерфейса. Однако, вместе с тем, Storyboard часто может быть громоздким и сложным для работы в больших проектах.

Swift - это язык программирования, разработанный компанией Apple. Он является основным языком для разработки приложений под платформы iOS, macOS, watchOS и tvOS. Swift обладает мощными возможностями и простым синтаксисом, что делает его более доступным для разработчиков. С помощью Swift можно создавать логику приложения, обрабатывать данные, взаимодействовать с API и многое другое.

SwiftUI - это новый фреймворк для создания пользовательских интерфейсов, представленный Apple в 2019 году. SwiftUI позволяет создавать пользовательские интерфейсы с помощью декларативного подхода. В отличие от Storyboard, SwiftUI не использует визуальное представление интерфейса, а вместо этого предлагает использовать компактный и удобный для работы код. Он упрощает создание и модификацию интерфейса, а также обеспечивает более гибкую адаптацию для различных устройств и ориентаций. SwiftUI также прямо интегрируется с языком Swift, что делает его удобным и эффективным инструментом для разработки приложений.

Ответ на вопрос о том, что лучше - Storyboard, Swift или SwiftUI, зависит от конкретных требований проекта и предпочтений разработчика. Storyboard может быть удобным для быстрого создания интерфейса визуально, особенно для начинающих разработчиков. Swift является мощным и гибким языком программирования, который подходит для сложной логики приложения. SwiftUI, с другой стороны, предлагает современный и упрощенный подход к разработке интерфейса. Он может быть предпочтительнее для создания новых приложений или для переноса существующих проектов на новейшую платформу.

Литература.

1.Р.Ш. Зайнулин, А.А. Бегалиева Вестник КРСУ.2022.Том 22.№2

2.https://apptractor.ru/info/articles/evolyutsiya-v-vashem-karmane-kak-razvivalis-mobilnyie-prilozheniya.html