1. История вопроса. Ранние мобильные устройства и используемые в них технологии. Наладонники. Первые смартфоны. iPhone и возникновение экосистемного подхода. Android как открытая платформа. Windows Phone и закат Nokia.

2. Мобильные операционные системы. Общий обзор структуры и функций ОС. Специфические требования, связанные с мобильностью. ОС Symbian – общая структура, слои API, организация асинхронного выполнения, активные объекты, графический интерфейс

3. Мобильные ОС (продолжение). ОС Android, iOS

4. Технологии мобильной разработки. Нативные среды разработки (iOS, Android, JavaME, Windows Phone). Используемые языки программирования, средства эмуляции и отладки, выгрузка на устройство или в магазин, сертификаты безопасности

5. Кроссплатформенные технологии мобильной разработки. Технологии, базирующиеся на HTML5. Гибридные технологии на примере одной из распространенных систем (PhoneGap, Cordova, Ionic и т.д.)

6. Кроссплатформенные технологии мобильной разработки. Xamarin. Конструкторы приложений «без программирования»

7. Кроссплатформенные технологии мобильной разработки. Технология UbiqMobile

8. Мобильные приложения. Архитектура и основные категории мобильных приложений. Специфика мобильных устройств с точки зрения сценариев использования и usability приложений. «Локальные» приложения и сервисы. Мобильный телефон как сервер для работы с более мелкими устройствами. Основные типы и категории мобильных приложений.

9. Социальные приложения. Web 2.0 как структурный подход к построению информационных систем. Скорость роста и формирование экосистемы. Критическая масса. Мобильные аспекты социальных приложений. Важность дизайна UI и usability для успеха социального проекта. Примеры.

10. Приложения для бизнеса – I. Традиционные приложения и сервисы для бизнеса. Электронная почта. Мессенджеры. «Корпоративные» приложения. Мобильное рабочее место Большого Начальника. Поддержка полевых сотрудников. Специфика требований бизнеса. BYOD и кроссплатформенность. Телефоны/планшеты, поддержка разных форм-факторов. Проблемы интеграции с базовыми информационными системами.

11. Приложения для бизнеса – II. Направленность «внутрь» и «вовне». Какие задачи нужно мобилизовывать, а какие нет? Интеграция с «материнскими» системами. Облако или выделенный сервер? Проблемы защиты и безопасности. Необходимость промежуточного сервера. Примеры архитектур различных систем.

12. Приложения для бизнеса – III. Настраиваемые мобильные сервисы. Примеры – документооборот, Интернет-торговля и др.

13. M2M и интернет вещей. Протоколы, используемые в интернете вещей.

1. История вопроса. Ранние мобильные устройства и используемые в них технологии. Наладонники. Первые смартфоны. iPhone и возникновение экосистемного подхода. Android как открытая платформа. Windows Phone и закат Nokia.

Первой попыткой вполне можно считать телефон IBM Simon, который был впервые представлен в качестве концепта в 1992 году. Стоимость устройства составляла около $900 с контрактом и примерно $1100 без него. Кроме функций телефона, аппарат получил функции органайзера, был способен отправлять и получать факс, позволял работать с электронной почтой, а также содержал несколько игр. Клавиши управления отсутствовали, а все действия совершались посредством сенсорного экрана. Из-за довольно больших габаритов и веса (0.5 кг) значительного распространения Simon не получил.

Hewlett-Packard вместе с Nokia в самом начале 1996 года выпустили коммуникатор 700LX – усовершенствованную версию HP 200LX с возможностью подключения мобильного телефона Nokia 2110. Для синхронизации работы с сотовым телефоном программное обеспечение было обновлено. Но данное устройство нельзя считать коммуникатором как таковым, поскольку оно состояло из двух частей, независимых друг от друга.

Не проходит и года, как Nokia выпускает Nokia 9000 Communicator – устройство, которое смогло объединить коммуникатор и мобильный телефон в одно. Данный аппарат получил:

- QWERTY-клавиатуру;
- Операционную систему GEOS (без возможности установки приложений);
- Черно-белый экран достаточно высокого разрешения по тем временам;
- Размеры 173 x 65 x 38 мм, вес около 400 г.

Не обошлось и без недостатков:

- Невозможность установки ПО;
- Отсутствие расширяемого функционала.

Со временем линейка КПК серии 9xxx была заменена на 9000i, 9110 и 9110i.

В следующем году была образована компания под названием High Tech Computer Corporation (HTC Corporation), целью которой стала разработка мобильных аппаратов, которые могли совместить функционал телефона и коммуникатора.

В 1998 году для разработки программного обеспечения для мобильных устройств был основан консорциум Symbian. В него вошли такие компании, как Psion, Motorola, Nokia и Ericsson.

Тем временем компания Nokia вполне уверенно чувствовала себя на рынке коммуникаторов. В 1999 году для рынка США вышли модели под названием Qualcomm pdQ 800 и pdQ 1900. Однако особой популярностью они не пользовались из-за больших размеров, веса и цены.

В 2000 году вышел новый телефон компании Ericsson под названием R380s. Именно его производитель назвал смартфоном. Среди особенностей R380s были:

- Относительно малый вес и габариты;
- Наличие сенсорного экрана;
- Откидывающаяся крышка.

Несмотря на название «смартфон», таковым его нельзя было назвать, хотя бы потому, что он не позволял устанавливать сторонние приложения.

Бурное развитие смартфонов и коммуникаторов началось в 2001 году: компания Nokia выпускает очередную модель коммуникатора: Nokia 9210. Данная модель работала под управлением Symbian OS 6.0 и была первым устройством серии 9xxx с открытой ОС. Устройство было основано на новой платформе series 80, несовместимой с программами для предыдущих поколений коммуникаторов Nokia. Модель обладала весьма внушительной функциональностью, внутренний экран был цветным. Кроме того был анонсирован телефон Nokia 7650, который считается первым «настоящим» смартфоном, поскольку он работал под управлением открытой для сторонних разработчиков операционной системы Symbian OS 6.1. Однако, компания Nokia позиционировала данную модель в первую очередь как имиджевый телефон с расширенными мультимедийными функциями, а не как интеллектуальное устройство с открытой ОС. Впрочем, маленький

размер доступной памяти и отсутствие разъёма для карты памяти сильно ограничивали возможности аппарата. В том же году появились первые коммуникаторы под управлением Pocket PC 2000.

В 2002 году выходит целый ряд коммуникаторов на базе платформы HTC Wallaby и смартфонов на базе платформы HTC Canary под разными торговыми марками. Коммуникаторы и смартфоны на базе операционных систем Microsoft получают значительное распространение и становятся массовыми. Неофициальное название «коммуникатор» закрепляется за с сенсорным экраном. В том же году появились смартфоны BlackBerry с QWERTY-клавиатурой, ориентированные на работу с электронной почтой. В то же время выпускается целый ряд коммуникаторов на базе Palm OS, которые стали весьма популярными. Коммуникаторы Nokia из-за высокой цены так и остались нишевыми устройствами.

В 2003 году компания Microsoft выпускает операционную систему Windows Mobile 2003. Nokia представляет сразу несколько смартфонов под управлением Symbian OS, а ряд производителей — под управлением Windows Mobile. Рынок смартфонов начинает стремительное развитие. В том же году выходит первый смартфон на платформе UIQ: Sony Ericsson P800. Данная модель продолжила ряд телефонов Ericsson с сенсорным экраном, закрываемым флипом, но уже с полным правом носила название «смартфон». В конце 2003 года компания Nokia также анонсировала свой первый сенсорный аппарат — Nokia 7700 на базе новой программной платформы series 90. Устройство должно было выйти в середине 2004 года, однако, после нескольких переносов срока выпуска было отменено и в широкую продажу не попало. Вместо него в конце года была выпущена модель Nokia 7710 — первый и единственный смартфон на рынке на базе series 90.

В 2005 году компания Microsoft выпустила Windows Mobile 5.0, обладающую целым рядом значительных улучшений. Операционная система была выпущена в трёх вариантах: для смартфонов, для КПК и коммуникаторов. Все три версии ОС были построены на единой платформе, этот факт способствовал увеличению функциональности смартфонов и размыванию границ между смартфонами и коммуникаторами. Компания Nokia объявила о прекращении поддержки программных платформ series 80 и series 90 и концентрации усилий на развитии единой для всех устройств платформы series 60. В конце 2005 года были представлены первые смартфоны Nokia на обновлённой платформе series 60 version 3

В 2006—2007 годах рынок смартфонов испытывает значительный рост. В начале 2006 года тайваньская компания High Tech Computer Corporation приняла решение о ликвидации торговой марки Qtek и продвижении своей продукции под единым брендом HTC. В этом же году компания выпустила HTC MTeoR — первый в мире 3G смартфон на базе Windows Mobile. Компания Palm, Inc. объявляет о начале сотрудничества с Microsoft и анонсирует коммуникатор Treo 700w под управлением Windows Mobile 5.0.

Компания Nokia выпускает линейку смартфонов на базе новых Symbian OS 9.1 и Symbian OS 9.2. Целый ряд устройств получил функции, характерные ранее только для коммуникаторов, обновленная программная платформа позволила осуществить поддержку больших разрешений. Кроме того, некоторые устройства обладали QWERTY/ЙЦУКЕН-клавиатурой. Линейка коммуникаторов Nokia была продолжена аппаратом Nokia E90. Одной из главных особенностей данной модели является то, что в сложенном виде она представляет собой полноценный смартфон. Все смартфоны N-серии в официальных пресс-релизах Nokia именуются «мультимедийными компьютерами». На базе обновленной UIQ 3 был выпущен ряд сенсорных смартфонов Sony-Ericsson. Кроме того, в 2007 году появился первый смартфон на базе UIQ 3.1 без сенсорного экрана Motorola Z8.

В первой половине 2007 года компания Microsoft выпустила Windows Mobile 6. Из названия версий операционной системы были исключены слова Smartphone и Pocket PC. Это окончательно закрепило объединение смартфонов и коммуникаторов в один класс устройств. Версия 6.0 отличалась от предыдущей в основном переработанным интерфейсом и рядом незначительных улучшений. Новая операционная система была совместима с программами для предыдущей версии.

В середине 2007 года компания Apple выпустила бесклавиатурный аппарат iPhone. Аппарат не отличался функциональностью, единственным аппаратным новшеством был способ управления устройством двумя пальцами. Однако емкостной экран, невиданный до той поры тактильный пользовательский интерфейс, создающий ощущение управления изображением на экране движениями пальцев, и агрессивная рекламная кампания сделали это устройство хитом продаж. Следует отметить, что изначально операционная система iPhone была закрытой, среда разработки приложений iPhone SDK для сторонних разработчиков появилась только в начале 2008 года. Смартфон от Apple привлек значительное внимание, многие производители

выпустили телефоны и коммуникаторы с интерфейсом, ориентированным на управление пальцами. Часто анонс таких аппаратов освещался в прессе как появление «убийцы iPhone».

В конце 2007 года компанией Google была анонсирована открытая мобильная платформа Android, основанная на ядре Linux, и была сформирована группа компаний Open Handset Alliance, целью которой является разработка открытых стандартов для мобильных устройств.

В середине 2008 года компания Google объявляет об открытии исходных кодов Android.
Компания Nokia также объявила о намерении открыть исходный код Symbian OS и начинает процесс покупки полного пакета акций Symbian, с целью образования некоммерческой организации Symbian Foundation. Процесс покупки был завершен 2 декабря 2008 года. Предполагается, что унифицированная открытая платформа Symbian появится в 2010 году.

В 2008 году Apple представила обновленную версию своего смартфона iPhone 3G. В аппарате появилась поддержка сетей 3-го поколения, GPS и исправлены некоторые недостатки предыдущей модели. Официальные поставки устройства осуществляются более чем в 70 стран. Благодаря успешным продажам iPhone компания Apple начала завоевание рынка смартфонов.

Осенью 2008 года вышел первый аппарат на базе Android — T-Mobile G1.

В конце 2008 года компании Sony Ericsson и Motorola отказались от дальнейшей разработки платформы UIQ. В это же время компания Nokia выпускает сенсорный аппарат Nokia 5800 на базе Symbian OS 9.4. Смартфон поддерживает управление без использования стилуса и ориентирован на массовый рынок. Одновременно с данным устройством был анонсирован флагманский смартфон Nokia N97 с сенсорным экраном и выдвижной QWERTY/ЙЦУКЕН-клавиатурой, который вышел в середине 2009 года.

2. Мобильные операционные системы. Общий обзор структуры и функций ОС. Специфические требования, связанные с мобильностью. ОС Symbian – общая структура, слои API, организация асинхронного выполнения, активные объекты, графический интерфейс

Функции ОС:

- Управление процессами
- Управление памятью
- Управление внешними устройствами (ввод-вывод)
    - Драйверы внутренних устройств
    - Работа с сетью и беспроводными соединениями
    - Управление подключаемыми устройствами
    - Поддержка специфических устройств (сканер отпечатков пальцев, сенсорный экран, акселерометр и т.д.)
- Защита и контроль прав доступа
- Организация постоянной памяти (файловая система)
- Графическая подсистема

3 уровня программных компонент ОС:

- Ядро
- Системные сервисы и утилиты
- Пользовательские программы и сервисы

Основные функции ядра ОС:

- Управление процессами, диспетчеризация, межпроцессное взаимодействие
- Обработка прерываний
- Управление памятью
- Защита и безопасность
- Драйверы устройств
- Работа с сетью
- Файловая система

Отличительные особенности ОС реального времени:

- Предсказуемость поведения в наихудшем случае
- Гарантированное время выполнения задач
- Надежность (отсутствие «зависаний», адекватное реагирование на сбои аппаратуры и сети)
- Автоматическое восстановление (перезапуск) после серьезных сбоев
- Жесткие приоритетные схемы при управлении процессами и задачами
- Нетребовательность к ресурсам, быстрая загрузка, «легкость» самой ОС
- Ориентация на профессиональных разработчиков, а не на широкий круг пользователей

Требования к мобильным ОС:

- Сочетание удобства использования и «гламурного» UI, ориентированного на непрофессиональных пользователей, с жесткими требованиями систем реального времени
- Относительно низкие требования к процессору и памяти
- Поддержка широкого набора телефонных и сетевых протоколов
- Экзотические устройства (GPS, акселерометр, гироскоп, камера и т.д.)
- Сенсорный экран
- Жесткие ограничения по энергопотреблению
- Жесткий real-time во всем, что относится к обеспечению GSM- и сетевых протоколов

Symbian

История берет начало в 1980 с системы EPOC компании Psion, в 2001г анонсирован Symbian v.6.0, до 2008 расцвет платформы и выход v.7.0, в 2011 Nokia переходит от Symbian в пользу Windows Phone, а в 2013 и вовсе от него отказывается.

ОС состоит из 5 основных уровней:

- Пользовательский интерфейс
- Сервисы приложений
    - PIM: личная информация, часовой пояс, календарь, контакты
    - Сообщения: почта и SMS
    - Интернет (встроенный браузер)
    - Синхронизация данных
- Сервисы ОС
    - Общие сервисы ОС
        - Управление процессами и потоками
        - Работа с таймером
        - Активные объекты
        - Работа с файлами
    - Графика и мультимедиа
    - Коммуникации (телефония, работа с сетью и беспроводными соединениями)
    - Взаимодействие с PC
- Базовые сервисы
- Ядро и взаимодействие с железом

Популярная на тот момент JavaME работала на первых двух вышеописанных уровнях

Элементы графического интерфейса

- Статусная панель (информация о заряде, сети, запущенных приложениях)
- Панель работающего в данный момент приложения
- Панель управления (кнопки “Выбор”, “Отмена”, ...)

Активные объекты и асинхронное выполнение

Способы организации асинхронного выполнения:

- Реальный параллелизм на основе процессов
- Реальный параллелизм на основе потоков
- Асинхронный параллелизм на одном потоке

Активные объекты – основные действующие лица:

- Выполняются на основном потоке приложения:
    - Собственно активный объект – интерфейс с асинхронным сервисом (аналог Listener)
    - Диспетчер активных объектов (Active Scheduler)
- Выполняются на параллельном потоке
    - Внешний асинхронный сервис (ASP – Asynchronous Service Provider)

```
// Создаем интерфейсный объект таймера – наследник CActive
RTimer aTimer;
// Привязываем созданный таймер к службе времени нижнего уровня
aTimer.CreateLocal();
// Устанавливаем интервал срабатывания таймера. Описываем переменную типа TRequestStatus и передаем
ее таймер. Автоматически она будет установлена в значение KRequestPending до момента наступления
таймерного события.
TRequestStatus timerStatus;
aTimer.After(timerStatus, 3000000);
// Выполняем какую-то работу, пока не истекло время ожидания
while(timerStatus == KRequestPending) { ... }
// Когда TRequestStatus получает значение, отличное от
KRequestPending, таймер завершен. KErrNone
показывает, что все завершилось без ошибок.
if (timerStatus == KErrNone) { _LIT(KOut, "Timer expired"); console->Write(KOut); }
// Очищаем таймер
aTimer.Close();
```

3. Мобильные ОС (продолжение). ОС Android, iOS

Архитектура Android (сверху вниз)

Источник: https://habrahabr.ru/company/intel/blog/184708/

Уровень приложений (Applications)

В состав Android входит комплект базовых приложений: клиенты электронной почты и SMS, календарь, различные карты, браузер, программа для управления контактами и много другое. Все приложения, запускаемые на платформе Android написаны на языке Java.

Уровень каркаса приложений (Application Framework)

Android позволяет использовать всю мощь API, используемого в приложениях ядра. Архитектура построена таким образом, что любое приложение может использовать уже реализованные возможности другого приложения при условии, что последнее откроет доступ на использование своей функциональности. Таким образом, архитектура реализует принцип многократного использования компонентов ОС и приложений.

Основой всех приложений является набор систем и служб:
1. Система представлений (View System) – это богатый набор представлений с расширяемой функциональностью, который служит для построения внешнего вида приложений, включающий такие компоненты, как списки, таблицы, поля ввода, кнопки и т.п.

2. Контент-провайдеры (Content Providers) – это службы, которые позволяют приложениям получать доступ к данным других приложений, а также предоставлять доступ к своим данным.
3. Менеджер ресурсов (Resource Manager) предназначен для доступа к строковым, графическим и другим типам ресурсов.
4. Менеджер извещений (Notification Manager) позволяет любому приложению отображать пользовательские уведомления в строке статуса.
5. Менеджер действий (Activity Manager) управляет жизненным циклом приложений и предоставляет систему навигации по истории работы с действиями.

Уровень библиотек (Libraries)

Платформа Android включает набор C/C++ библиотек, используемых различными компонентами ОС. Для разработчиков доступ к функциям этих библиотек реализован через использование Application Framework. Ниже представлены некоторые из них:
1. System C library — BSD-реализация стандартной системной библиотеки C (libc) для встраиваемых устройств, основанных на Linux.
2. Media Libraries – библиотеки, основанные на PacketVideo‘s OpenCORE, предназначенные для поддержки проигрывания и записи популярных аудио- и видео- форматов (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG и т.п.).
3. Surface Manager – менеджер поверхностей управляет доступом к подсистеме отображения 2D- и 3D-графических слоев.
4. LibWebCore – современный движок web-браузера, который предоставляет всю мощь встроенного Android-браузера.
5. SGL – движок для работы с 2D-графикой.
6. 3D libraries – движок для работы с 3D-графикой, основанный на OpenGL ES 1.0 API.
7. FreeType – библиотека, предназначенная для работы со шрифтами.
8. SQLite – мощный легковесный движок для работы с реляционными БД.

Уровень среды исполнения (Android Runtime)

В состав Android входит набор библиотек ядра, которые предоставляют большую часть функциональности библиотек ядра языка Java.

Платформа использует оптимизированную, регистр-ориентированную виртуальную машину Dalvik, в отличии от нее стандартная виртуальная машина Java – стек-ориентированная. Каждое приложение запускается в своем собственном процессе, со своим собственным экземпляром виртуальной машины. Dalvik использует формат Dalvik Executable (*.dex), оптимизированный для минимального использования памяти приложением. Это обеспечивается такими базовыми функциями ядра Linux, как организация поточной обработки и низкоуровневое управление памятью. Байт-код Java, на котором написаны ваши приложения, компилируются в dex-формат при помощи утилиты dx, входящей в состав SDK.

Уровень ядра Linux (Linux Kernel)

Android основан на ОС Linux версии 2.6, тем самым платформе доступны системные службы ядра, такие как управление памятью и процессами, обеспечение безопасности, работа с сетью и драйверами. Также ядро служит слоем абстракции между аппаратным и программным обеспечением.

Архитектура IOS(снизу вверх)

источник: https://xakep.ru/2014/10/08/kau-ustroena-ios/

Darwin

Условно начинку iOS можно разделить на три логических уровня: ядро XNU, слой совместимости со стандартом POSIX (плюс различные системные демоны/сервисы) и слой NeXTSTEP, реализующий графический стек, фреймворк и API приложений. Darwin включает в себя первые два слоя.
По своей сути Darwin — это «голая» UNIX-подобная ОС, которая включает в себя POSIX API, шелл, набор команд и сервисов, минимально необходимых для работы системы в консольном режиме и запуска UNIX-софта.

XNU
Ключевой компонент Darwin - гибридное ядро XNU, основанное на ядре Martch и компонентах ядра FreeBSD, таких как планировщик процессов, сетевой стек и виртуальная файловая система.

UNIX/BSD
Уровнем выше ядра в Darwin располагается слой UNIX/BSD, включающий в себя набор стандартных библиотек языка си (libc, libmatch, libpthread и так далее), а также инструменты командной строки, набор шеллов (bash, tcsh и ksh) и демонов, таких как launchd и стандартный SSH-сервер.
На этом открытая часть ОС под названием Darwin заканчивается, и начинается слой фреймворков.

Frameworks
Darwin реализует лишь базовую часть iOS, которая отвечает только за низкоуровневые функции (драйверы, запуск/остановка системы, управление сетью, изоляция приложений и так далее). Та часть системы, которая видна пользователю и приложениям, в его состав не входит и реализована в так называемых фреймворках — наборах библиотек и сервисов, которые отвечают в том числе за формирование графического окружения и высокоуровневый API для сторонних и стоковых приложений

В стандартной поставке iOS можно найти десятки различных фреймворков, которые отвечают за доступ к самым разным функциям ОС — от реализации адресной книги (фреймворк AddressBook) до библиотеки OpenGL (GLKit). Набор базовых фреймворков для разработки графических приложений объединен в так называемый Cocoa Touch API, своего рода метафреймворк, позволяющий получить доступ к основным возможностям ОС.

SpringBoard и BackBoard
Уровнем выше находятся приложения, и центральное SpringBoard.

SpringBoard — это связующее звено между операционной системой и ее пользователем, графический интерфейс, позволяющий запускать приложения, переключаться между ними, просматривать уведомления и управлять некоторыми настройками системы

Разделен на две части, сам рабочий стол и сервис BackBoard, который отвечает за коммуникацию с низкоуровневыми слоями архитектуры.

Applications
На вершине архитектуры находятся приложения, различаемые на стоковые (высоко привилегированные) и на сторонние.

4. Технологии мобильной разработки. Нативные среды разработки (iOS, Android, JavaME, Windows Phone). Используемые языки программирования, средства эмуляции и отладки, выгрузка на устройство или в магазин, сертификаты безопасности

Технологии мобильной разработки:

1. Нативная
2. Кроссплатформенная

Рассмотрим плюсы нативной разработки:

1. Скорость работы приложения. Т.к. приложение создается с использованием оригинальных инструментов разработки, получаемый в результате компиляции проекта код является оптимальным для данной платформы.
2. Гибкость в реализации.
3. Использование последних технологий
4. Легкость и качество тестирования.
5. Полная поддержка со стороны магазинов приложений App Store и Google Play

Платформа: iOS

Нативные среды разработки: xCode на MacOS
ЯП: Objective-C, Swift
Эмуляция и отладка:
Apple iOS Simulator входит в Xcode
Под Windows **iPadian**
Выгрузка:

1. iOS Developer Program — 99$ в год
2. iOS Enterprise Developer Program — 299$ в год (нельзя выкладывать в магазин, но можно разрабатывать in-house – приложения для внутреннего использования в своей компании, без открытого доступа)
3. Ad Hoc — это способ распространения приложений без AppStore, прямой установкой файла-сборки приложения через iTunes. (не более 100 устройств, и приложение работает не более 6 месяцев)

Сертификаты безопасности:

Платформа: Android

Нативные среды разработки: Eclipse, Android Studio
ЯП: Java, C++, Delphi
Эмуляция и отладка:
Встроенные отладчики Android Studio и Eclipse
Выгрузка:
Google Play – разовая плата 25$
Samsung Apps Store
Amazon Apps Store
Сертификаты безопасности: a

Платформа: JavaME

Нативные среды разработки: Eclipse
ЯП: Java
Эмуляция и отладка:
Sun Java Wireless Toolkit, mpowerplayer
Выгрузка:

Нет централизованного магазина. Можно бесплатно выкладывать на 4pda
Сертификаты безопасности: a

Платформа: WinPhone

Нативные среды разработки: Visual Studio
ЯП: C#
Эмуляция и отладка: в IDE
Выгрузка:
Windows Store:
Частное лицо – 49$ в год
Компания – 99$ в год
Студентам – бесплатно
Сертификаты безопасности:

5. Кроссплатформенные технологии мобильной разработки. Технологии, базирующиеся на HTML5. Гибридные технологии на примере одной из распространенных систем (PhoneGap, Cordova, Ionic и т.д.)

Идея
Один код = несколько приложений для разных платформ. Пишем адаптивный сайт с необходимым функционалом, заворачиваем в обертку -> приложение для нужной платформы готово.

Преимущество использование html5 в мобильной разработке:

- Лёгкое вхождение для веб-разработчиков
- Дешево в разработке
- Большое покрытие (браузер сейчас есть везде)
- Единая база кода

Недостатки:

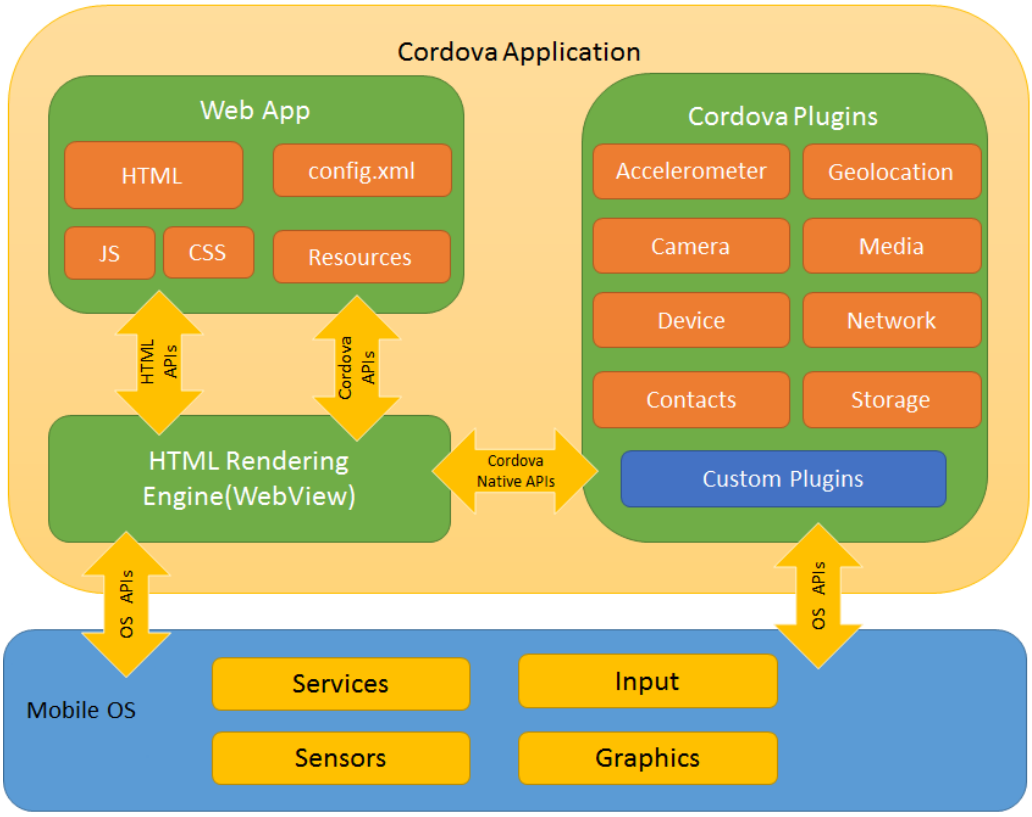
- Производительность UI - Манипуляции с UI работают в один поток, большое потребление памяти, т.к. кроме кода приложения необходимо запускать WebView, скорость исполнения JavaScript невысока, работа с DOM потребляет много ресурсов.
- Нет унификации платформ - Доминирует WebKit, но существует Windows Phone, в котором необходимо поддерживать IE. Каждая платформа предоставляет свои размеры экранов для телефонов и планшетов. Разные требования к UI на каждой из платформ.
- Чужеродность UI - Приложение - всего лишь HTML-страница. Sencha Touch, jQuery Mobile не похожи на нативные приложения.
- Отсутствие средств разработки и отладки “из коробки”
- Ограниченный доступ к аппаратным возможностям - Набора плагинов иногда бывает недостаточно

Когда использовать:

- Когда хочется поддерживать единый код.
- Когда необходимо иметь приложение, охватывающее весь спектр платформ.
- Когда нет жестких требований к UI, он простой и не содержит сложных эффектов.

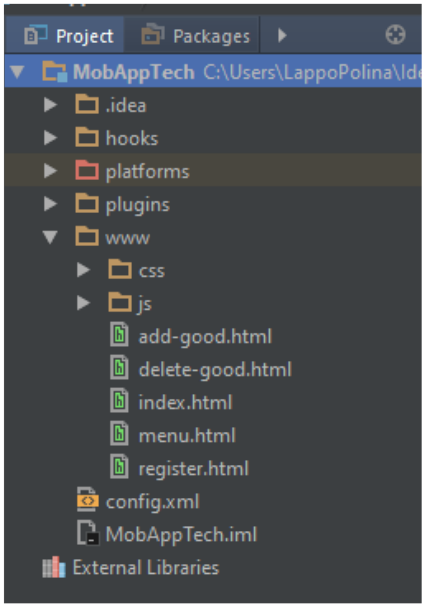
Apache Cordova
Фреймворк Apache Cordova позволяет создавать мобильные приложения для различных платформ используя стандартные веб-технологии.
Чтобы понять, как устроено приложение Cordova, следует рассмотреть его основные компоненты:

- config.xml – общий файл для всех платформ, который содержит информацию о приложении и определяет параметры, влияющие на то, как приложение работает;
- HTML, CSS, JavaScript, Resources (Web App) – само приложение реализовано как набор html-страниц, которые ссылаются на CSS, JavaScript, изображения, файлы мультимедиа или другие ресурсы необходимы для запуска;
- WebView – браузер в пределах мобильной платформы, который позволяет отображать пользовательский интерфейс приложения;
- Cordova Plugins – обеспечивает доступ к нативным компонентам устройств.



Пикча с оф сайта

Структура Cordova проекта в IDE



Весь код приложения, который вы пишете находится в папке **www**. Внутренняя структура папки настраивается по вашему усмотрению.

В папку **plugins** добавляются плагины. На странице каждого из плагинов приведена консольная команда для его установки, а также описаны методы и примеры использования.
Cordova Plugin (Plugin) – дополнительный код, который обеспечивает для JavaScript доступ к нативным компонентам устройств.

Папка **platforms** содержит подпапки платформ для которых будет собираться приложение. В этой же папке и будет выходной файл для установки на мобильное устройство. Например путь до apk файла с андроидовсим приложением *platforms\android\build\outputs\apk*

Файл конфигурации config.xml
файл содержит различные настройки необходимые для работы приложения. Из основных это:

- Названия приложения
- пусть к начальной странице приложения
- разрешение доступа к интернету
- Разрешения к к различным приложения устройства (камера, файловая система, контакты и смс)

Также в этом файле указывается информация об авторе предложения.

6. Кроссплатформенные технологии мобильной разработки. Xamarin. Конструкторы приложений «без программирования»

Конструкторы приложений без программирования

Некоторая часть мобильный приложений составляет из себя однотипные приложения, например: служба такси, служба заказа пиццы, служба доставки, бронирования столиков и пр.
В большинстве своем данные приложения не отличаются по детальному функционалу, а отличаются только по дизайну и набору “фич”, например, раздел “новости” или “чат с оператором”.
Для решения задачи построения таких приложений существуют онлайн сервисы (платные), позволяющие выбрать тип приложения, набор “фич” (модули) и дизайн, настроить необходимые данные (названия, куда отправлять данные и пр.) и на выходе получить готовые собранные файлы, готовые для загрузки в магазины приложений.

Преимущества:

- Крайне быстрый старт
- Готовая, протестированная функциональность
- Дизайны, разрабатываемые специалистами

Недостатки:

- Нет возможности самостоятельно добавить дополнительную функциональность
- Платные тарифы работы с подобными системами
- Нет возможности изменить дизайн (можно сменить только логотипы, экраны загрузки, цвета)

Xamarin

Xamarin позволяет создавать одну единственную логику приложения с применением C# и .NET сразу для всех трех платформ - Android, iOS, Windows Mobile. То есть Xamarin представляет технологию для кроссплатформенной разработки мобильных приложений.

Преимущества:

- В процессе разработки создается единый код для всех платформ
- Xamarin предоставляет прямой доступ к нативным API каждой платформы
- При создании приложений мы можем использовать платформу .NET и язык программирования C# (а также F#)

Функционально платформа Xamarin представляет ряд субплатформ. В частности:

- Xamarin.Android - библиотеки для создания приложений на ОС Android
- Xamarin.Mac - библиотеки для создания приложений на Mac OS X
- Xamarin.iOS - библиотеки для создания приложений для iOS

Эти субплатформы играют большую роль - через них приложения могут направлять запросы к прикладным интерфейсам на устройствах под управлением ОС Android или iOS.

Благодаря этим платформам мы можем создавать отдельно приложения для Android, отдельно для iOS, но наиболее важной особенностью платформы является возможность создавать кроссплатформенные приложения - то есть одна логика для всех платформ. То есть мы один раз можем определить визуальный интерфейс, один раз к нему привязать какую-то логику на C#, и все это будет работать на Android, iOS и Windows Phone.

Xamarin.Forms

Отдельный тип PCL приложений, использующих одноименную библиотеку.
Позволяет писать один UI и логику поведения на все платформы.
Приложения Xamarin.Forms работают на Universal Windows Platform (Windows 8+)

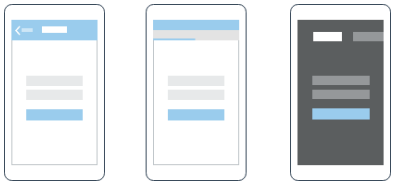
Преимущества:

- Во-вторых, подход к созданию и работе с пользовательским интерфейсом близок к тому, к чему мы все привыкли в Windows. Особенно рады будут разработчики WPF, так как Xamarin Forms поддерживает работу с XAML, биндинги, темплейты, стили и прочие радости жизни. Думаю, понятно, что они несколько урезаны и не стоит ожидать всей мощи WPF, но все-таки удобства это добавляет.
- Из того, что Xamarin.Forms схож с WPF, вытекает следующий плюс этой платформы: MVVM. Действительно, Xamarin.Forms имеет XAML, визуальные элементы имеют BindingContext (аналог DataContext в WPF), есть BindableProperty (аналог DependencyProperty). Таким образом, можно связывать View с ViewModel аналогично тому, как в WPF.
- Еще одно преимущество данной платформы в том, что так как UI описывается только в одном месте, то приложения под разными системами будут выглядеть очень похоже. Что может быть важно, например, в корпоративных разработках.

Недостатки:

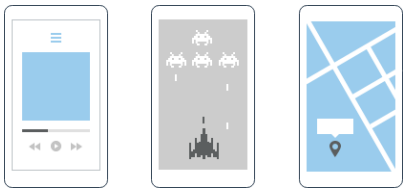
- Неполная реализация функционала WPF
- Компромиссные решения в реализации функционала, различающегося на разных платформах
- Различное поведение на разных платформах
- Производительность

Что выбрать



Для чего хорош Xamarin.Forms?

- Приложения для работы с данными
- Proof-of-concept / prototype
- Приложения с небольшими платформенными зависимостями
- Для приложений, когда переиспользование кода важнее кастомизации и нативности интерфейса

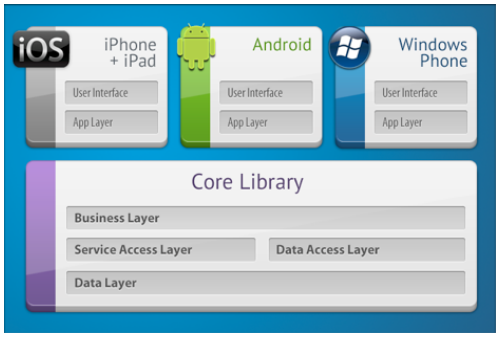


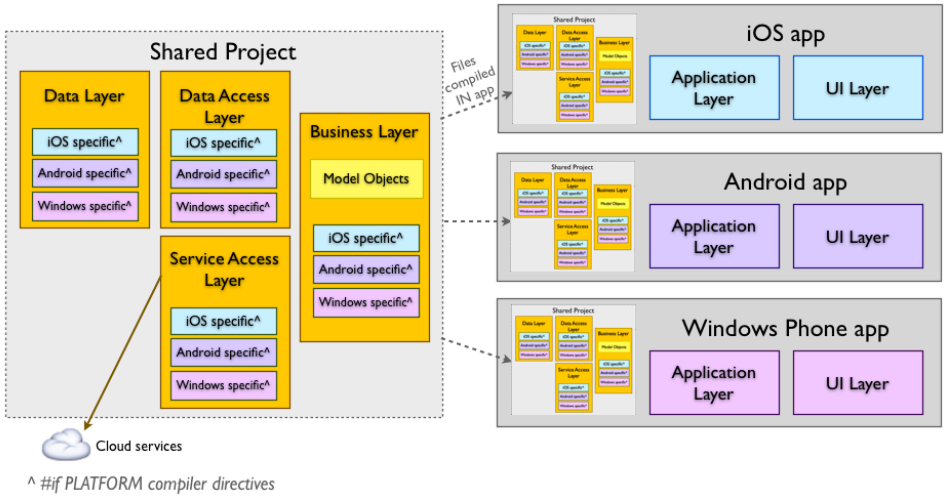
Для чего хорош подход с разработкой отдельных UI?

- Для приложений, которым нужны особые варианты взаимодействий
- Для приложений с точным дизайном
- Для приложений с большими платформенными зависимостями
- Для приложений, когда нативность и кастомизация интерфейса важнее переиспользования кода

Альтернативный путь

Помимо технологии Xamarin.Forms (одно приложение, 3 платформы) и просто Xamarin (3 приложения на 3 платформы) есть “промежуточный” подход, при котором мы выносим всю функциональность не связанную с интерфейсом (работа с сетью, камерой, локальной БД) в отдельную библиотеку, а в платформо-зависимых приложениях мы оставляем только код отвечающий за интерфейс.

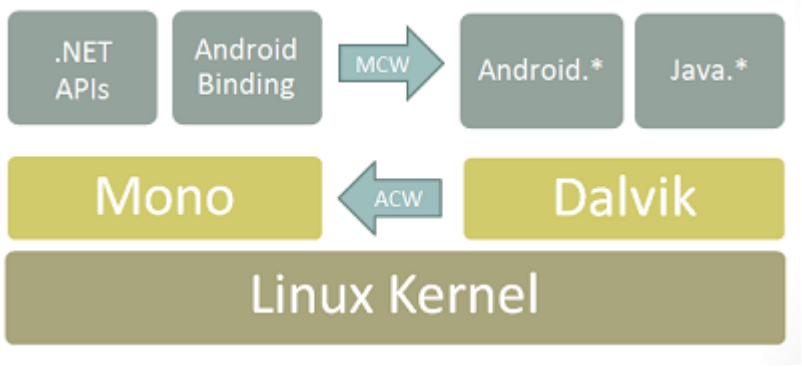




Что под капотом

Для iOS ситуация простая — никакой виртуальной машины нет и программный код должен быть просто заранее скомпилирован в машинный. Для этой цели используется AOT компилятор Mono.

Для Android интереснее. При компиляции приложения происходит перевод кода на C# в промежуточный байт-код, понятный виртуальной машине Mono и сама эта виртуальная машина также добавляется в упакованное приложение. И Mono и Dalvik написаны на Си и работают поверх ядра Linux. При запуске приложения на Android обе виртуальные машины начинают работать бок о бок и обмениваются данными через специальный механизм wrapper`ов.

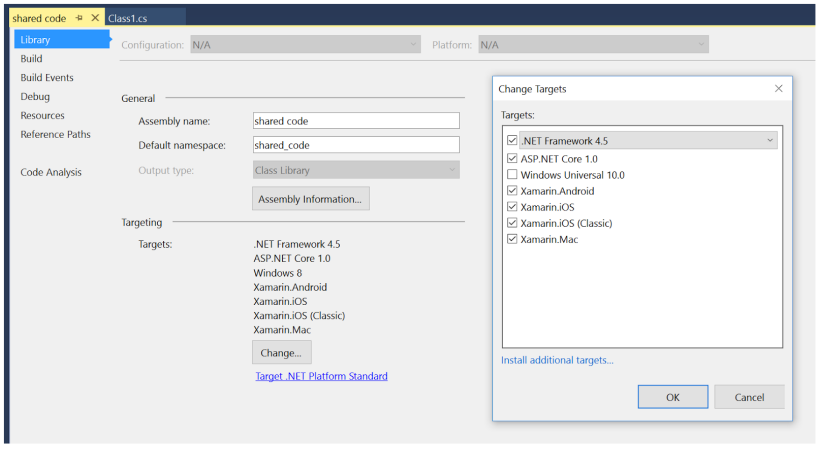


Сторонние компоненты

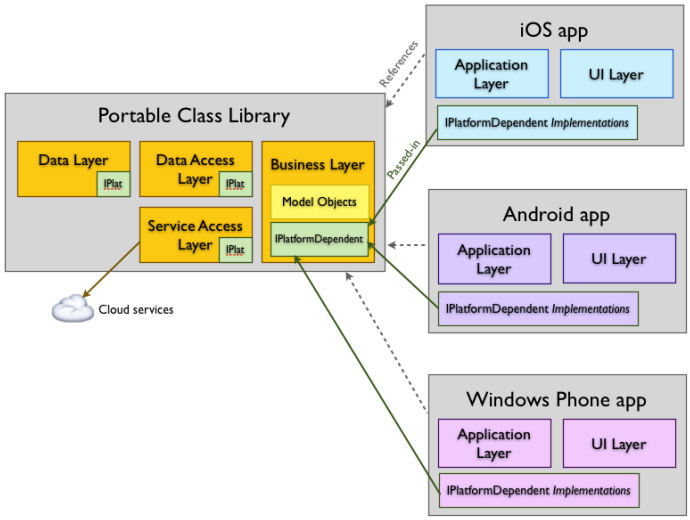
У Xamarin существует собственный магазин сторонних компонентов Xamarin Components (много хороших бесплатных/платных компонентов, например сканирование/генерация штрих-кодов).
Помимо этого можно подключать PCL пакеты из менеджера пакетов NuGet.

PCL (Portable Class Library) пакеты

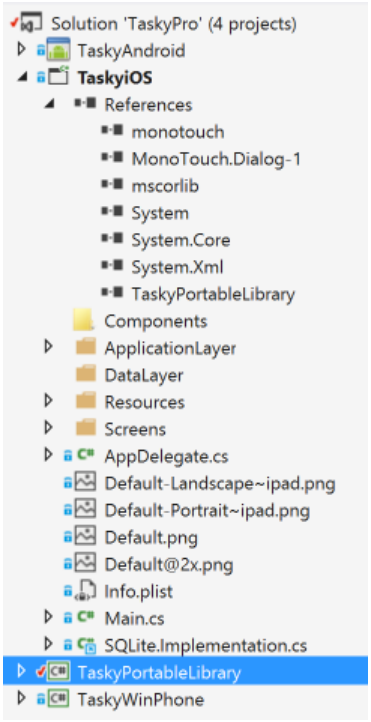
Для того, чтобы при работе с внешними пакетами или при вынесении части логики не собирать DLL под каждую из платформ (при использовании платформенно-зависимого функционала) можно собирать платформонезависимые библиотеки, которые запускаются на определенном круге платформ (настраивается в библиотеке).



Также PCL библиотеки могут объявлять набор интерфейсов, который необходим для работы, которые само приложение, которое встраивает PCL библиотеку должно реализовывать (Dependency Injection).



Пример проекта с выносом общего кода в PCL библиотеку TaskyPortableLibrary



Дополнительно

Разработка Xamarin поддерживается в Visual Studio (можно даже разрабатывать под Windows, а сборку под iOS осуществлять удаленно на отдельной Mac станции). Помимо этого присутствует кроссплатформенная IDE Xamarin Studio.

Изначально Xamarin был платным фреймворком с бесплатной версией для обучения и инди-проектов, но, после того, как его выкупил Microsoft, он стал бесплатным.

7. Кроссплатформенные технологии мобильной разработки. Технология UbiqMobile

UbiqMobile — кросс-платформенная технология, позволяющая создавать сложные мобильные приложения для делового использования. Она позволяет быстро и эффективно разрабатывать и публиковать мобильные приложения и осуществлять их сопровождение в течение всего жизненного цикла. Использование UbiqMobile позволяет быстро и эффективно «мобилизовать» бизнес-процессы предприятий, предоставляя сотрудникам удобный доступ к корпоративной информации и расширяя

аудиторию конечных пользователей и клиентов. Результатом является повышение доступности предоставляемых продуктов и сервисов для конечных пользователей, рост лояльности клиентов и, в конечном счете, повышение эффективности бизнеса.

Основные особенности UbiqMobile:

- **Кросс-платформенность**. Единожды разработанное приложение работает на различных мобильных платформах, не требуя доработки под конкретные типы телефонов.
- **Простота разработки**. Для разработки не требуется навыков мобильного программирования; вся разработка ведется в единой среде на базе .NET.
- **Легкость модификации «на лету»**. Можно менять бизнес-логику и данные приложений на стороне сервера «на лету», не затрагивая и не переустанавливая мобильные приложения пользователей.
- **Безопасность**. Вся бизнес-логика приложения и все существенные данные хранятся на отдельном сервере и практически не оставляют информации на конечном мобильном устройстве (телефон, планшет).
- **Неприхотливость к качеству интернета**. Приложения UbiqMobile работают в любых условиях, в том числе на медленных и не всегда устойчивых соединениях, в регионах с недостаточно развитой инфраструктурой мобильной связи, в сельской местности и т.д
- **Высокая интегрируемость**. Приложения легко интегрируются с существующими информационными системами.
- **Экономичность**. Расходы на Интернет при работе с приложениями UbiqMobile существенно ниже, чем при обычном Интернет-серфинге
- **Масштабируемость**. Серверная часть приложений может размещаться на сервере в облаке, что позволяет при необходимости легко наращивать ресурсы.

Среда разработки (UbiqMobile SDK)

- Среда разработки UbiqMobile представляет собой расширение (Plug-in) для Microsoft Visual Studio
- Включает в себя IDE, среду времени выполнения и набор клиентских компонент для различных мобильных платформ
- Мощные и удобные инструментальные средства, входящие в состав IDE, облегчают дизайн, разработку, отладку и тестирование приложений
- Набор предустановленных шаблонов и библиотечных компонент избавляет разработчика от значительного объема рутинной работы по реализации типовой бизнес-логики приложений
- Тестирование приложений возможно как в режиме эмуляции, так и на реальных телефонах. До 90% всей работы по тестированию может быть выполнено в режиме эмуляции
- Серверная часть готового приложения выгружается в облако или на выделенный сервер «в один клик»

Разработка, отладка и публикация приложений происходит в рамках единой среды разработки, что существенно повышает эффективность и сокращает сроки разработки.

Мобильные клиенты

Для создания мобильных приложений для устройств, работающих под управлением систем Android, iOS, Windows Phone, а также Java ME, вам не потребуется опыт мобильного программирования для этих платформ. Все приложения, созданные с использованием UbiqMobile, могут выполняться в окне универсального клиента-контейнера – подобно тому, как в обычном веб-браузере можно открыть любую страницу HTML.

Для запуска приложений UbiqMobile с любой поддерживаемой мобильной платформы вы можете либо использовать готовый универсальный мобильный клиент-контейнер, либо создать отдельное мобильное приложение, оформленное в соответствии с фирменным стилем вашей компании.

Универсальный клиент-контейнер можно установить из соответствующего магазина приложений – в частности, Google Play (для Android-устройств) и Windows Store (для Windows Phone-устройств). Для установки на Java-телефоны доступна WAP-ссылка. Когда на вашем устройстве установлен универсальный клиент, все разрабатываемые вами приложения становятся доступны сразу же после выгрузки на сервер, не требуя ожидания публикации приложения в соответствующем магазине.

Для публикации приложений UbiqMobile в Apple AppStore необходимо использовать индивидуализированные версии клиентов. Все, что для этого нужно сделать – это настроить необходимые параметры (логотип, цвета, заставки, режимы по умолчанию) при помощи специального мастера дизайна клиентских приложений, входящего в состав инструментальных средств UbiqMobile, и послать запрос на удаленный сервис сборки, который скомпонует клиентские модули специально для вашего приложения и пришлет вам набор исполняемых файлов. После этого можно публиковать ваше приложение в Apple AppStore!

Сферы использования приложений UbiqMobile

UbiqMobile является универсальной системой, можно выделить несколько основных категорий приложений, для создания которых UbiqMobile будет особенно эффективна:

- Мобильные расширения информационных систем предприятий
- Социальные сети, онлайн-доски объявлений и другие подобные массовые Интернет-сервисы
- Интерактивные системы доступа к информации
- Системы удаленного доступа к «интеллектуальным» устройствам (видеокамеры, системы «Умный дом» и т.д.

8. Мобильные приложения. Архитектура и основные категории мобильных приложений. Специфика мобильных устройств с точки зрения сценариев использования и usability приложений. «Локальные» приложения и сервисы. Мобильный телефон как сервер для работы с более мелкими устройствами. Основные типы и категории мобильных приложений.

[АРХИТЕКТУРА]

Существует три подхода технической реализации приложений для мобильных устройств:

Мобильное native-приложение — это специально разработанное приложение под конкретную мобильную платформу (iOS, Android, Windows Phone). Такое приложение разрабатывается на языке высокого уровня и компилируется в т. н. native-код ОС, обеспечивающий максимальную производительность. Минусы - требуют от разработчика специальных знаний и умений для работы в конкретной среде разработки (xCode для iPhone, eclipse для устройств на Android); низкая переносимость между мобильными платформами, трудоемкий процесс разработки. Такие приложения с легкостью могут использовать все функции ПО смартфона (камера, микрофон, акселерометр, геолокация, адресная книга, плеер и т.д.), и при этом более бережно расходуют ресурсы телефона (аккумулятор, память). В зависимости от назначения приложения предполагают или не предполагают наличие интернет-соединения. Пример – Instagram.

Мобильное web-приложение — специализированный web-сайт, адаптированный для просмотра и функционирования на мобильном устройстве. Пользовательский интерфейс создается при помощи стандартных веб-технологий. Их не нужно загружать из магазина приложений, но они могут

находиться в специальных магазинах веб-приложений, которые есть у некоторых современных браузерах, например у Chrome.Главной особенностью таких приложений является их кроссплатформенность — возможность работать на всех устройствах, без дополнительной адаптации. Независимо от установленной ОС такие приложения не могут использовать ПО смартфона. Для обновления информации в приложении необходимо подключение к интернету, скорость работы ограничена возможностями интернет-соединения провайдера услуг. При желании продавать приложение вам потребуется собственная платежная система. Пример – Google Maps.

Гибридное приложение — мобильное приложение, "упакованное" в native-оболочку. Гибридные приложения сочетают в себе некоторые функции нативных и веб-приложений: кроссплатформенность и возможность использования ПО телефона. Такие приложения могут быть загружены через магазины приложений, и при этом имеют возможность независимого обновления информации. Гибридные приложения требуют подключения к интернету, поскольку веб часть обновляется через интернет. Несколько уступают в производительности нативному. Это, наверное, самый популярный способ построения мобильных приложений, так как у него органическая среда распространения, но разработка происходит быстрее и дешевле, чем в случае с нативными приложениями, так как, хотя оболочка и написана на «родно» языке программирования, «начинка» может быть написана в том или ином объеме на html5. Пользователь же скорее всего не заметит разницу между нативным приложением и гибридным. Пример – HeartCamera для iOS

Таблица преимуществ и недостатков

Нативное	Максимальная функциональность и скорость работы Не требуется интернет- соединение для использования Имеет доступ к ПО смартфона (GPS, плеер, камера) Распространение через магазины приложений	Выше стоимость и длиннее сроки разработки Требует от разработчика знаний определенной среды программирования Работает только с одной платформой При косметических изменениях необходимо выпускать обновление
Веб (HTML5)	Кроссплатформенность Не требует загрузки из магазина мобильных приложений Можно легко адаптировать обычный сайт Легче найти веб-разработчика нежели разработчика под определенную платформу Простота создания и поддержки	Требует подключения к интернету Не имеет доступа к ПО смартфона Не может отправлять push-уведомления Должен быть запущен интернет-браузер При продаже требуется использование своей платежной системы
Гибридное	Функциональность нативного приложения на независимой платформе Запускается не из браузера в отличии от веб приложения	Загружается из магазина мобильных приложений (необходимо соответствовать требованиям) Разработчик должен быть знаком с разными API

	Возможность независимого обновления Распространение через магазины приложений	

На всякий случай еще это здесь оставлю.. https://habrahabr.ru/post/246877

[ТИПЫ И КАТЕГОРИИ + СПЕЦИФИКА]

Существует множество классификаций МП. Если обобщить, то можно выделить несколько их типов:

- ***промо-приложения*** (выступают в качестве поддержки рекламных кампаний): Axe Hot Fever
- ***приложения-события*** (организация мероприятий): Eventum, Event Advisor
- ***приложения-службы*** (являются своеобразными аналогами сайтов): Яндекс.Карты
- ***приложения-игры***
- ***интернет-магазины***: Lamoda
- ***МБ для бизнеса***: Booking, Office Mobile
- ***социальные сети***
- ***другие*** (контентные приложения, системные приложения)

При разработке приложения необходимо учитывать специфику устройства, под которое это приложение разрабатывается, с точки зрения usability. Главные отличия приложения, разрабатываемого под планшет и смартфон от десктопной версии – отсутствие устройства ввода первых и функциональная составляющая (приложения и веб-сайты, как правило, призваны решать совсем разные задачи). Например, при разработке мобильного приложения необходимо учитывать следующее:

- простой дизайн
- крупные элементы
- небольшое количество элементов на экране
- несколько экранов
- преобладание графических элементов над текстовыми

Приложения под планшеты и смартфоны обладают, как правило, меньшим функционалом. Они предназначены для использования с целью быстрого доступа (например, к почте). При этом не включают дополнительные опции, которыми пользователь, скорее всего, воспользуется с десктопной версии сайта, так как это более удобно.

[ЛОКАЛЬНЫЕ ПРИЛОЖЕНИЯ]

Локальные МБ - приложения, работающие в режиме оффлайн (карты, читалки, игры и др.).

Принципы разработки offline-first приложения (с хабра аж 12 года):

- максимальная отвязка приложения от сервера
- объект-обёртка для серверного API на стороне клиента
- отвязка обновления данных от хранилища данных

[МОБИЛЬНЫЙ ТЕЛЕФОН КАК СЕРВЕР]

Современные (и не только) мобильные устройства (преимущественно, на базе Android) могут выступать в качестве серверов. Для того, чтобы поднять сервер у себя на телефоне, необходимо обладать правами администратора. Это может быть веб-сервер, файловый, FTP-сервер, медиасервер и др.

Любой правильно настроенный сервер получит локальный IP-адрес, но у всей вашей сети, выходящей в интернет, будет единый внешний IP-адрес. Если в планах использовать сервер только в пределах домашней сети, то для подключения к нему потребуется знание локального IP-адреса. Кроме того, для правильной работы серверов может понадобиться переназначение портов (port forwarding) на вашем домашнем роутере: такое перенаправление позволит всем устройствам в сети «видеть» порт, через который работает сервер.

Для того чтобы использовать Android-сервер не только в рамках локальной сети, но и за её пределами, необходимо воспользоваться услугой динамического DNS либо оплатить статический IP-адрес и обзавестись доменным именем.

Для создания сервера на базе устройства под управлением Android можно воспользоваться специализированными программами, которые, как обычно, загружаются с Google Play. Примеры: Servers Ultimate, My FTP Server и Pixel Media Server.

9. Социальные приложения. Web 2.0 как структурный подход к построению информационных систем. Скорость роста и формирование экосистемы. Критическая масса. Мобильные аспекты социальных приложений. Важность дизайна UI и usability для успеха социального проекта. Примеры.

Социальное приложение - Социальные сети. Удобны для быстрого общения и обмена информацией, просмотра новостей и уведомлений. Существует приложения для глобальных сетей, а также для узких и брендированных, например, соцсети BMW и Adidas.

Социальные приложения на смартфонах – это логическое продолжение самой идеи телефона как средства связи. Однако социальные приложения позволяют не ограничивать эту связь звонками. Человек является социальным существом, и это находит отражение в том, что даже такие "несоциальные" приложения как, скажем, браузер, имеют социальные функции, например "поделиться ссылкой". И возможности общения этим не ограничиваются, так как практически любое социальное взаимодействие можно перенести на смартфон, то есть, фактически, поместить пользователю в карман.

О каких программах идет речь? Прежде всего, это:

• Клиенты социальных сетей (Одноклассники, Фейсбук, VK).

• Приложения для новых знакомств в сети Интернет.

• Программы для накрутки лайков в известных социальных сетях.

WEB 2.0 (определение Тима О'Рейлли) — методика проектирования систем, которые путём учёта сетевых взаимодействий становятся тем лучше, чем больше людей ими пользуются. Особенностью веб 2.0. является принцип привлечения пользователей к наполнению и многократной выверке информационного материала.

Определение Тима О'Рейлли нуждается в уточнении. Говоря «становятся лучше», имеют в виду скорее «становятся полнее», то есть речь, как правило, идёт о наполнении информацией, однако вопросы её надёжности, достоверности, объективности не рассматриваются.

По сути, термин «Web 2.0» обозначает проекты и сервисы, активно развиваемые и улучшаемые самими пользователями: блоги, вики-проекты, социальные сети и т. д.

Подробная статья об экосистемах: https://www.osp.ru/os/2014/02/13040044/

Краткая выжимка по билету:

Часто экосистема образуется из связанных проектов и технологий, многие из которых сначала развивались в рамках проекта, а впоследствии стали самостоятельными приложениями или продуктами. Изначально принцип экосистем использовали несколько компаний, которые совмещали разработку программного и аппаратного обеспечения. Пример среди телефонов: Nokia, Siemens, Ericsson и Motorola. С появлением смартфонов инвестиции в развитие как аппаратного, так и программного обеспечения значительно выросли, и такие компании сделали программные платформы доступными внешним разработчикам, создав начальную экосистему и вернувшись к своей основной деятельности по проектированию и дизайну оборудования.

Следующее поколение экосистем: программные платформы, подходящие для любых устройств. Принципиально экосистемы отличаются возможностью изменения платформ.

Экосистема процветает, когда компания может предложить своим клиентам широкий выбор продуктов, удовлетворяющих все их потребности. Экосистема позволяет распределять усилия между участниками. Будучи в экосистеме, компании могут эффективно распределить все функции управления разработкой программного обеспечения.

Принципы экосистемы: предоставление большего выбора и расширение текущего предложения для уже имеющихся клиентов и пользователей;

- повышение привлекательности для новых клиентов и пользователей, снижение затрат на модернизацию функциональности путем деления расходов на техническое обслуживание и прочие непрофильные функции с другими участниками экосистемы;
- ускорение внедрения новаций в экосистеме за счет более динамичной обратной связи от участников;
- формирование новых принципов программно-аппаратного взаимодействия и разработки универсальных масштабируемых платформ для предоставления более широкого спектра программных услуг.

Эффект критической массы - экосистемы может развиваться и успешно функционировать только набрав критическую массу пользователей. При ее отсутствии ее поддержка может оказаться фатальной и не эффективной. (Примеры: Windows Phone, Apple Pay)

Это происходит по следующим причинам:

- Часть партнеров экосистемы это сообщество экосистемы, пользователи и фирмы поставляющее для нее продукцию
- Пользователи создают обратную связь для экосистемы, что позволяет быстрее удовлетворять их потребности
- В современном мире очень сложно предоставить полноценный функционал вне экосистемы (особенно на рынке мобильных устройств).
- Учитывая скорость инноваций, цикл продукта можно определить в два года — компания, которая за это время не выведет на рынок продукт, отвечающий или предупреждающий потребности клиента, быстро теряет рынок.

Мобильные аспекты соц. приложений - использование геолокации, push-уведомления, live-трансляции и т.д.

В идеале приложение для мобильного устройства должно работать со скоростью мысли. Более того, интерфейс приложения должен быть понятен даже новичку.

10. Приложения для бизнеса – I. Традиционные приложения и сервисы для бизнеса. Электронная почта. Мессенджеры. «Корпоративные» приложения. Мобильное рабочее место Большого Начальника. Поддержка полевых сотрудников. Специфика требований бизнеса. BYOD и кроссплатформенность. Телефоны/планшеты, поддержка разных форм-факторов. Проблемы интеграции с базовыми информационными системами.

В качестве бизнес – инструмента мобильные приложения помогают достичь следующих целей компании:

1. Снижение издержек
2. Прирост прибыли
3. Оптимизация внутренних процессов (усиление контроля за всеми бизнес – процессами)
4. Улучшение качества предоставляемых услуг
5. Привлечение новых клиентов (новые каналы связи)
6. Повышение статуса, авторитетности и компетентности компании в глазах клиента

Приложения/сервисы для бизнеса:

1. Облачные хранилища данных (Dropbox, Google drive, Bitrix24)
2. Удаленный рабочий стол (Chrome)
3. Приложения – помощники для поиска информации и выполнения различных рабочих операций (Office Mobile, планировщики, сканеры документов...)
4. Справочники (Консультант Плюс – содержит федеральное законодательство, финансовые консультации...)
5. Новости (Zite – статьи по выбранному кругу интересов)

Электронная почта – основной инструмент бизнес-коммуникаций:

1. Чтение, редактирование, отправка и пересылка писем
2. Автоматическое получение уведомлений о новых письмах
3. Просмотр, скачивание и отправка вложений
4. Поиск, группировка и удаление писем
5. Инструменты управления корреспонденцией (фильтрация, работа с “важностью”)
6. Интеграция с планировщиками и другими инструментами управления временем

Мессенджеры (Viber...)

В основе:

1. Асинхронность
2. Долговременность
3. Реальность диалогов
4. Распространенность и повсеместность

По мобильному номеру клиента (мобильный обычно найти проще, чем адрес электронной почты)

“Корпоративные” приложения

1. Позволяют собирать отчетность
2. Получать информацию непосредственно на гаджеты
3. Ставить задачи и контролировать их выполнение

4. Организовывать совещания
5. Быстро передавать, просматривать, загружать, редактировать и делиться корпоративными данными

При этом пользователи соблюдают корпоративные политики безопасности по работе с данными, что повышает их сохранность при удаленной работе.

Мобильное рабочее место большого начальника

Решение, которое помогает решать управленческие задачи. Совокупность функциональных модулей, объединенных в одно приложение (документооборот, отчетность, календарь...)

Задача – предоставление руководителям возможности работать практически в любом месте, без привязки к офису. Функционал формируется в зависимости от индивидуальных потребностей и специфики работы.

Выгоды:

1. Оперативное принятие управленческих решений
2. Свобода перемещений и возможность участия в большом количестве деловых встреч
3. Ускорение согласования и подписания важных документов

Мобильные технологии помогают компании наладить эффективную работу внештатных сотрудников или сотрудников в различных регионах (**полевые сотрудники**).

1. Возможность видеть местонахождение сотрудника (по GPS), историю перемещений; отслеживание времени
2. Собирать данные на месте (показатели датчиков, мониторинг цен…), дневной отчет, опросы, аудит
3. Фиксировать объекты на фото/видео
4. Планирование визитов/задач (построение маршрутов)

Важными требованиями к мобильным приложениям являются простота использования и возможность работы офлайн.

BYOD (bring your own device) – подход к организации рабочего места сотрудника, при котором он применяет принадлежащее ему устройство для доступа к информационным ресурсам компании.

Плюсы:

1. Экономия бюджета
2. Эффективность (комфорт сотрудников)
3. Лояльность (повышение уровня доверительного отношения между компанией и сотрудником)

Риски:

1. Устройство может быть утеряно или украдено (вместе со всей корпоративной информацией)
2. Зараженные планшеты в офисе могут нанести вред другим сотрудникам через рабочую сеть

Кроссплатформенность

Плюсы:

1. Экономия бюджета

2. Время разработки
3. Поддержка и обновление продукта
4. Единая логика приложения

Минусы:

1. Медленная работа приложения
2. Не используются уникальные особенности платформы
3. Непривычный для пользователя интерфейс

Форм-фактор – размеры экрана и типы устройств (телефон, планшет…)

Основной проблемой при разработке дизайна для мобильных устройств является наличие нескольких форм-факторов.

Например – Windows 8 и Windows Phone.

Раньше приходилось использовать разделяемую переносимую библиотеку для выделения общего кода, отвечающего за доступ к данным и бизнес-логику, и различные пакеты для UI.

Сегодня существуют возможности расширения приложений до универсальных (работающих независимо от форм-фактора устройств). Некоторые аспекты пользовательского интерфейса приложения будут автоматически адаптированы под все устройства (кнопки, ползунки…).

Интеграция с базовыми информационными системами

Проблемы:

1. Повторный ручной ввод данных (справочники, финансовые транзакции…)
2. Многократные и бесконечные “сверки и корректировки”, не исключающие ошибок
3. Непомерные затраты на формирование сводной отчетности
4. Неприемлемые сроки и себестоимость выполнения даже обыденных задач

Цели интеграции:

1. Уменьшить стоимость эксплуатации совокупности приложений предприятия
2. Увеличить скорость выполнения типичных задач или гарантировать сроки их выполнения
3. Поднять качество выполнения задач за счет формализации процессов и минимизации человеческого фактора, как основного источника ошибок

11. Приложения для бизнеса – II. Направленность «внутрь» и «вовне». Какие задачи нужно мобилизовывать, а какие нет? Интеграция с «материнскими» системами. Облако или выделенный сервер? Проблемы защиты и безопасности. Необходимость промежуточного сервера. Примеры архитектур различных систем.

Направленность «внутрь» и «вовне».

Ахтунг! Надежной информации по этому пункту не найдено, поэтому далее следуют догадки.

Приложения, «направленные внутрь» ориентированы на внутренние процессы в компании. Их задачи: оптимизировать бизнес-процессы, повысить эффективность работы, снизить затраты (временные и денежные), улучшить коммуникацию. Примеры: справочники, мессенджеры и почта, Help Desk, планировщики, корпоративные приложения, профильные приложения для курьеров, логистов, мерчендайзеров, HR-менеджеров, руководителей и других.

Приложения, «направленные вовне» ориентированы на взаимодействие с клиентами. Это могут быть: приложения для поиска новых клиентов и установления контактов, приложения для

поддержки клиентов, для повышения лояльности, приложения для заказа продукции клиентом (дополнительный канал продаж).

Какие задачи нужно мобилизовывать, а какие нет?

В статьях о трендах мобильных приложений за 2016-2017 год упоминается, что ранее компании в большей степени использовали мобильные приложения, направленные вовне: искали дополнительный канал продаж, пытались повысить лояльность аудитории. В последних трендах то, что компании начинают использовать приложения для внутренних нужд, оптимизации бизнес-процессов. Вывод: для себя компании решили, что им нужно мобилизовать некоторые внутренние процессы.

И снова ахтунг! Дальше много оценочных суждений автора. В кавычках – цитаты из статей.

В первую очередь мобильные приложения должны быть удобны в использовании, а не усложнять работу. Какой бы навороченный смартфон ни был, у него имеются свои ограничения, то есть, размер экрана и памяти, удобство ввода данных. Приложения должны быть по возможности легковесными, с максимально простым интерфейсом, с минимумом ввода с клавиатуры.

На заметку: «В мобильной разработке фокус смещается на микроприложения, которые позволяют сотрудникам получить доступ к определенной функции корпоративного решения на смартфоне. Мобильность является ключевой бизнес-инициативой, которая помогает бизнесу быть гибче и для клиентов, и для сотрудников.»

«В основе создания корпоративных приложений должен лежать следующий принцип: приложение должно не просто выполнять свои задачи и быть удобным в использовании, оно должно обеспечивать возможность совместной работы, обмена информацией, должно быть простым в развертывании и поддержке – по возможности пользователь должен уметь самостоятельно решать все свои вопросы, связанные с продуктом.»

«Мобилизовывать» может быть целесообразно:

- Приложения с каталогом продукции и возможностью заказа.

  Нужно иметь в виду при разработке приложения для клиентов:

  С точки зрения экономии ресурсов самым предпочтительным вариантом выглядит web разработка.

Если планируется онлайн работа проекта как основной вариант взаимодействия с пользователем – безусловно, надо начинать с сайта, который может охватывать не только мобильных клиентов, но и пользователей стационарных компьютеров. В случае успеха можно далее реализовать отдельно мобильные приложения на выбранные платформы. Для большинства бизнес-приложений такой вариант наиболее подходит.

Если проект предусматривает больше оффлайн работу и нацелен на мобильных пользователей, то тут стоит отдать предпочтение приложениям.

Для реализации приложений, требующих высокой производительности интерфейса, дальновиднее использовать мобильные технологии, так как мобильное приложение наиболее тесно интегрировано с платформой и дает реализовать привычный отзывчивый интерфейс.

- Приложения с программами лояльности (скидки, баллы),
- Приложения для коммуникации (мессенджеры, почта),
- Планировщики (не столько для составления планов и расписания, сколько для получения уведомлений),
- Приложения с навигацией (например, для курьеров),
- Приложения для считывания информации с кодов (допустим, на складе),

- С натяжкой: профильные новостные ленты с заголовками,
- В скором времени возможно (судя по трендам): приложения, интегрированные с IoT: интеграция приложений с IoT продолжит упрощаться, и компаниям из сфер здравоохранения, образования, автомобилестроения, умных домов и т.д. стоит начать экспериментировать и внедрять IoT функции.
- Приложения-агрегаторы. Это инструменты, позволяющие получить контент из разных ресурсов и объединить его в простом понятном интерфейсе. Это могут быть новости из разных источников, со временем подстраивающиеся под пользователя, приложения для сбора статистики или отчетов для руководителей.
- «Интерактивные push уведомления: Push-уведомления - одна из самых популярных функций в приложении. Мы ожидаем, что они будут основной движущей силой, помогающей компаниям взаимодействовать с целевой аудиторией. Через push пользователи узнают о выгодных предложениях, акциях и новостях, получают персональные предложения, не открывая приложение. Это экономит время и срок службы батареи. Чтобы взаимодействие было успешным для обеих сторон, уведомления станут более индивидуальными и интерактивными.»

Нецелесообразно:
- Громоздкие корпоративные приложения (опять-таки, сейчас все любят микро, однофункциональные решения),
- Что-нибудь, связанное с документооборотом (Приложения для создания документов и графиков) – неудобно читать, точные данные не ввести)
- Анкеты, опросы клиентов,
- …

Интеграция с «материнскими» системами.

*Вспомним, что для бизнес-приложений есть три варианта: веб-приложение, мобильное приложение и веб-приложение, включающее в себя компонент браузера (будем называть гибридным). Мобильное обычно содержит **подмножество** функций Web-приложения, приспособленных к выполнению на устройстве.*

- Интеграция с платформами: Мобильные приложения далеко опережают сайт. В приложении существенно больше возможностей для доступа к устройству. В гибридном такая разница нивелируется. Кроме того, постоянно растет уровень предоставления доступа к возможностям устройства из браузера через расширяющийся набор API.

- Для реализации проекта на всех или каких-то определенных платформах требуется разработать **мобильное приложение** для каждой из платформ отдельно, причем на каждой свои среда и язык разработки, свои стандарты интерфейса. В случае **веб** одна версия должна покрывать потребности всех платформ. *Так выглядит в теории*. Но на практике оказывается, что браузеры на различных платформах функционируют по-разному. Приходится поддерживать либо несколько версий одного сайта, либо в коде подстраивать выдаваемый контент под текущий запрос. Существенные отличия в размерах экрана также сказываются и на верстке сайта.

- *Если компания думает, что ей лучше использовать, веб или мобильное (ахтунг, субъективные суждения из интернетов)*: На разработку "родных" приложений для корпоративных **мобильных** пользователей потребуется много денег и времени. Для сложного приложения могут потребоваться несколько специалистов (сложнее найти), а проблемы совместимости платформ увеличат затраты еще больше. Помимо первоначальной разработки, ввиду непрерывных обновлений операционной системы потребуется выпуск новых версий приложения, которые необходимо заново проверять. **Web-приложения (здесь гибридные**

имеются в виду), основанные на HTML5, CSS3 и JavaScript™ или использующие клиентскую среду вроде Dojo и jQuery, предоставляют простой способ сделать Web-сайты дружественными для мобильных устройств. Затраты на их разработку обычно намного ниже, потому что программистов, знакомых со стандартными Web-интерфейсами, найти легче. Обновления ОС не так критичны

Облако или выделенный сервер?

Облако	Выделенный сервер
Гибкость: · Добавить/отказаться от ресурсов (процессорное время, оперативная память…) без необходимости замены аппаратных частей · Возможно автоматическое изменение ресурсов в зависимости от нагрузки, можно и по запросу (изменение тарифного плана) - вертикальное масштабирование ● горизонтальное масштабирование (можно получить практически любое количество виртуальных машин нужной конфигурации). · Тарификация по фактическому потреблению · Можно расширить конфигурацию · Любые сервисные работы (добавление дисков в серверы, добавление новых машин и т.п.) в «облаке» будут выполнены быстрее. Моментальная доступность ресурсов	Жесткое ограничение ресурсов · Сервер может быть использован не полностью, а платить за целый · Вместо вертикального масштабирования - заказать другой сервер и переехать на него · ограниченность ресурсов собственной конфигурацией, сложность расширения конфигурации при возникновении пиковых нагрузок. · сложнее горизонтальное масштабирование, сложнее обратное (отказ от ресурсов)
Легкость реализации	
если сервер находится в облаке, то процедура настройки параметров приложений несколько сложнее, чем для обычного сервера · Ограничения инфраструктуры (аппаратная часть, специфичное ПО)	Полный контроль: возможность установки/удаления и настройки параметров приложений.

отказоустойчивость выше возможность разместить свой веб-проект не на одном виртуальном сервере, а на веб-кластере, состоящем из нескольких резервирующих друг друга серверов, расположенных в разных датацентрах. Если же нет возможности (например, дорого) постоянно держать полную копию проекта, можно использовать значительно более слабую машину, например, под бэкап данных и базы (например, slave MySQL). В случае аварии проект на резервном сервере быстро переводится в рабочий режим, а сам сервер масштабируется до нужного количества доступных ресурсов (память, CPU и т.д.)	
Деньги: · на первый взгляд дешевле · При оплате «по потреблению» при резком росте нагрузки (DDoS, «хабраэффект», ошибки в разработке) возможны значительные расходы (в разы больше запланированных). · Почти всегда отдельно рассчитывается стоимость траффика · Нет инсталляционных платежей Для больших проектов – если речь идет о покупке собственного оборудования – больше не нужно выделять значительные средства из бюджета · Минимальные финансовые риски на старте нового проекта · Слишком дорого для небольших проектов («Не является альтернативой хостингу за 200-300 руб./мес») · Сложность расчетов по потреблению · Дополнительные плюшки, которыми так гордятся облака, стоят дополнительных денег (в том числе меры безопасности и прочие)	
Человеческие ресурсы для обслуживания системы: Если речь идет о большом парке машин (десятки-сотни-тысячи), то в виртуальной среде обслуживать их гораздо легче. Компания Microsoft рассказывала о собственном опыте: один системный администратор может обслуживать в «облаке» в десять (и более) раз больше машин, чем реальных физических серверов.	

Затраты (время) на обучение сотрудников специфике конкретного сервиса	Классика, удобно пользоваться «по старинке»

Проблемы защиты и безопасности.

КВД = критически важные данные пользователей
Основные виды атак на мобильное приложение:

- Декомпиляция файла приложения (.ipa-файлы для Apple iOS и .apk-файлы для Google Android) и разбор локально сохраненных данных. Защита этого, наиболее важного в настоящее время, уровня целиком лежит на плечах мобильного разработчика.
- Перехват данных, передаваемых по сети (MITM-атаки). Хотя современная мобильная и веб-разработка активно завершают переход на HTTPS-протокол общения, тем не менее, не стоит полагаться на единственный рубеж защиты в виде защищенного канала связи.
- Рутование устройства и атака на приложение и применяемые в нем алгоритмы через внешние отладочные инструменты.

Уязвимости:

- хранение КВД в незащищенных или слабо защищенных локальных хранилищах, специфических для конкретной платформы. Легко вскрывается.
- Хранение КВД в коде: хранение приватного ключа для асимметричных алгоритмов; хранение паролей и логинов для серверных узлов или баз данных. Легко вскрывается третьей стороной при наличии базовых навыков декомпиляции.
- Применение алгоритмов с хранением приватного ключа. Или ключ известен слабо защищенному серверу
- Использование самописных алгоритмов шифрования и защиты (нельзя изобретать велосипеды)
- Передача КВД во внешнюю среду в открытом (незашифрованном) виде (например, передача данных стороннему приложению или передача в сеть)
- Перевод части функционала во встроенные веб-движки (Чаще всего выглядит как передача КВД во встроенный браузер, где загружается внешняя веб-страница, выполняющая свою часть функционала. Не использовать браузер, если в операциях замешаны КВД!)

Еще пара полезных мыслей:

- Если при разработке приложения внутри компании используются некие собственные алгоритмы, которые могут представлять высокую ценность для потенциальных конкурентов или взломщиков, то эти алгоритмы должны быть защищены от постороннего доступа.
- Для клиент-серверных приложений очень полезно применять сессионный механизм с ограниченным временем жизни сессии. Это позволит избежать "простаивания" приложения в незащищенном режиме, если пользователь просто забыл закрыть его и оставил устройство в свободном доступе.

- В тренде - мобильный банкинг, финансовые приложения, бесконтактные платежи. Безопасность подобных приложений иногда вызывает сомнения. Забавно, что смартфон с банковским приложением привязан чаще всего к аккаунту для получения смс-кодов.

- При хищении аппарата у злоумышленника появится доступ к приложениям, где вход осуществляется лишь раз, в том числе: почта и мессенджеры, Dropbox и другие хранилища, документы, заметки (нередко с паролями), и другие

- Использование смартфона или планшета для удаленного доступа к рабочему месту посредством VNC, TeamViewer и прочих средств удаленного администрирования уже не редкость. Так же как и доступ к корпоративной сети через VPN. Скомпрометировав свое устройство, сотрудник может скомпрометировать всю «защищенную» сеть предприятия.

- Современные мобильные устройства и приложения ориентированы на использование множества облачных сервисов. Необходимо следить, чтобы конфиденциальные данные и данные, относящиеся к коммерческой тайне, не были случайно синхронизированы или отправлены в один из таких сервисов.

- Многие приложения не проверяют, зарутовано ли устройство. "Просмотрев почти все МБ (мобильный банкинг) на российском рынке, можно сказать, то что лишь некоторые из них (точнее всего 2-3) проверяют, в каком состоянии находится мобильное устройство – есть ли там jailbreak или root-доступ. Далее, они выводят сообщение и информируют пользователя об этом. Еще в прошлом году одно из таких приложений отказывалось работать на «дискредитированном» устройстве, но уже в этом году ПО просто вносит определенный лимит на проводимые операции. А в большинстве случаев приложения для МБ работают как ни в чем не бывало."

Необходимость промежуточного сервера.

Промежуточный сервер Используется в многозвенных клиент-серверных приложениях для интеграции распределенных компонентов в единую информационную систему. Компонент, являющийся дескриптором для всей интеграционной системы, безопасности, связи, масштабируемости, кроссплатформенности и т. д. Данный компонент не хранит данные, он только передает их из серверной системы на мобильное устройство и обратно. Большинство мобильных платформ также содержат мобильный конфигуратор, позволяющий компаниям-разработчикам создавать и настраивать мобильные приложения.

Промежуточное ПО представляет набор сервисов, обращение к которым позволяет различным приложениям, в общем случае выполняющимся на разных платформах, взаимодействовать между собой. Общие прикладные интерфейсы (API) промежуточного ПО позволяют реализовать взаимодействие между приложениями, не углубляясь в инфраструктуру и детали реализации гетерогенной сети, а последующие изменения в структуре и составе такой сети не потребуют изменений в приложениях (при условии, что эти изменения не затрагивают API middleware). Промежуточный сервер напоминает среднее звено в трехзвенных клиент-серверных архитектурах, за исключением того, что функциональные части middleware распределены между приложениями и/или их компонентами в корпоративной сетевой среде.

Сервисы middleware представляют приложениям разнобразные функции API, которые, в сравнении с функциями операционных систем и сетевых служб, обеспечивают:

- **прозрачный доступ к другим сетевым сервисам и приложениям;**
- **независимость от других сетевых сервисов;**
- **высокую надежность и постоянную готовность.**

Когда используется промежуточный сервер?

1. **важна кроссплатформенность приложений:**
2. **важна безопасность (может фильтровать данные как для удобства, так и из соображений безопасности)**
3. **нужно собирать информацию из различных подразделений, функциональных и географических**
4. **для осуществления быстрых и защищенных транзакций в различных компьютерных средах (e-commerce)**
5. **прозрачный доступ**
6. **надежность, бесперебойность работы**

Промежуточный сервер нужен в компаниях, которые используют мобильные приложения в одной инфраструктуре, которая масштабируется под различное число мобильных сотрудников и доступна в онлайн и офлайн режимах.

Примеры архитектур различных систем

12. Приложения для бизнеса – III. Настраиваемые мобильные сервисы. Примеры – документооборот, Интернет-торговля и др.

Можно выделить два основных подхода к поставке подобного рода приложений:

1. Разработчик выкладывает в маркет приложение, которое позже может сконфигурировано под нужды заказчика.
   Проливается вода по сабжу

   Такой интересен в первую очередь разработчикам систем документооборота.

   Преимущества:
   - таким образом проще доставлять обновления. Действительно, если мы говорим об общем критическом обновлении для всех (например, усправление уязвимости), то достаточно выложить одно обновленное приложение в маркет. Если мы говорим об обновлении для одного заказчика, то достаточно поправить конфигурацию. Обновление придет в обход маркета (+10 к скорости).

   - из предыдущего пункта вытекает в некотором смысле повышенный уровень надежности и безопасности (мы же можем быстрее реагировать на косяки)

   Недостатки:
   - Необходимость обратной совместимости. Заказчик использует старую схему конфигов и не хочет ничего менять.

   - => apk может быть сильно раздут.

   Это подход часто используют разработчики различных СЭД (SharePoint, EOS, 1c и др.)

2. Согласно требованиям заказчика на основе некотрого шаблона генерируется готовое приложение для мобильных устройств (например, apk для Android)
   Плещем рассказом о том, что все хотят собственное приложение

   Преимущества:
   - Личное полноценное приложение в маркете со всем вытекающим.
   - при правильном формировании apk, не содержит ничего лишнего.

   Недостатки:
   - усложненное обновление.

- когда у тебя залито куча приложений в маркете, можно запутаться.

Этот подход снискал популярность в сфере интернет-торговли (PRICE APP, FlipCat и др.)

3. Также стоит отметить еще один вариант, когда на основе шаблона делается мобильный сайт и устанавливается как приложение.

   Преимущества:
   - Возможность быстро доставлять обновления (еще быстрее, чем в первом варианте, ведь не нужно ждать валидации на маркете)

   - Проще устанавливать (немного спорный момент, потому что пользователи знают, как ставить приложения, но могут не знать, как сайт на главный экран выкатить, но это в инструкции можно описать)

   Недостатки:
   - Несмотря на то, что стандарт HTML5 многое привносит для того, чтобы webapp выглядел нативно, не все и не всё сейчас поддерживается (те же нотификации вроде только в Android есть).

   - Ограниченный объем выделенной памяти под приложение (могут возникнуть сложности с файлами (картинками) большого размера, если требуется поддержать работу без интернета).

   Очевидно, что такой подход будет интересен, если нужно сделать небольшое приложение, например, для какого-то события. Нужна легкость в установке и уметь очень быстро обновлять это приложение в случае форс-мажоров. В качестве примера можно рассмотреть идею сервиса LocalEvents.

13. «Локальное» использование мобильного телефона. NFC. Работа с дополнительными устройствами. M2M и интернет вещей. Протоколы, используемые в интернете вещей.

Локальное использование (без выхода в сеть Интернет) мобильного телефона:

1. IrDA (инфракрасный порт)
2. Bluetooth
3. NFC

IrDa - группа стандартов, описывающая протоколы физического и логического уровня передачи данных с использованием инфракрасного диапазона световых волн в качестве среды передачи.

Был популярен в 90е и начале 2000-х, но со временем был вытеснен, потому что:
1. в корпуса устройств нужно было монтировать ИК-прозрачное окно
2. Ограниченная дальность действия и прямая видимость между приёмником и передатчиком
3. Низкая скорость передачи данных на момент вытеснения (в современных версиях стандарта скорость весьма высока, но былая слава утрачена)

Варианты использования на мобильном устройстве: передача файлов (мультимедиа, контакты, и др.), синхронизация, сейчас - использование мобильного устройства как пульта дистанционного управления

Bluetooth

Технология Bluetooth является твердо устоявшимся коммуникационным стандартом для беспроводной связи на малых расстояниях, соединяя устройства посредством одной универсальной радиолинии с малым радиусом действия. Изначально дальность действия радиоинтерфейса закладывалась равной 10 метрам, однако сейчас спецификациями Bluetooth уже определена и вторая зона — около 100 м. При этом нет необходимости в том, чтобы соединяемые устройства находились в зоне прямой видимости друг друга. К тому же, взаимодействующие между собой приборы могут находиться в движении.

Представлена в 1994г., по-настоящему популярна стала в 2004г. после выхода Bluetooth 2.0

Актуальная версия - Bluetooth 5, представлена в 2016г, начинают выходить девайсы с поддержкой данного стандарта (например, Samsung Galaxy S8), основной вектор улучшений направлен на IoT (снижение энергопотребления, увеличение радиуса действия и др.)

Популярна и сегодня за счет довольно большой дальности действия. Варианты использования:

1. Передача файлов
2. Беспроводные устройства воспроизведения музыки (наушники, колонки), hands-free гарнитуры
3. **Bluetooth beacons - https://en.wikipedia.org/wiki/Bluetooth_low_energy_beacon** как часть M2M/IoT, умные дома\лампочки\утюги\и пр.
4. Подключение прочих периферийных устройств (мышь\клавиатура), носимых девайсов (фитнес-трекеры и пр.)

NFC

NFC - технология беспроводной передачи данных малого радиуса действия, которая дает возможность обмена данными между устройствами, находящимися на расстоянии около 10 сантиметров, т.е. в случае мобильных телефонов устройства нужно приложить друг к другу

Представлена впервые в 2004г, бум начался в 2011г. (начало внедрения в Android-смартфоны), в 2014 г. Apple также внедрила чипы NFC в свои смартфоны (в iOS возможно использование NFC только для эмуляции кредитной карты - см. https://ru.wikipedia.org/wiki/Apple_Pay)

Ключевые преимущества перед Bluetooth:

1. Ощутимо более короткое время установки соединения
2. Совместимость с существующими RFID-структурами (это означает, что второму устройству не обязательно иметь источник питания, например, кредитная карта с возможностью бесконтактной оплаты)

Недостатки

1. Меньшая скорость передачи данных
2. Меньший радиус действия

Получается, типичный use-case это “вытащил, приложил, пошел дальше”. Из этого вытекают варианты использования:

1. Использование телефона как кредитной карты
2. Запись и чтение NFC-меток - от использования в режиме хранения информации (электронная доска объявлений) до выполнения команд на телефоне при считывании метки (например, “перевести в беззвучный режим”)
3. Спаривание Bluetooth - для соединения устройств Bluetooth 2.1 и выше, поддерживающих NFC, достаточно сблизить их и принять соединение. Процесс активации Bluetooth с обеих сторон, поиска, ожидания, соединения и авторизации заменён простым «прикосновением» мобильных телефонов.

В ближайшем будущем:

1. Электронные билеты
2. Удостоверения личности
3. Ключи от машины, квартиры, итд
4. Предлагайте своё :)

Про дополнительные устройства -

1. можем подключать через USB используя OTG (https://en.wikipedia.org/wiki/USB_On-The-Go) - флешки, внешние диски, мышки, клавиатуры, джойстики
2. беспроводные технологии описаны выше - на сегодняшний день это
   2.1. NFC (там, где важна скорость инициации соединения, а количество передаваемых данных мало + короткая сессия - внешними устройствами выступают терминалы оплаты с поддержкой NFC и прочие считыватели, а также прочие мобильные устройства для инициации соединения по Bluetooth, например)
   2.2. Bluetooth - долгие сессии (вся периферия, что подключается по USB, сюда подходит), устройства IoT/M2M, носимые девайсы, гарнитуры/наушники
   2.3. в редком случае Wi-Fi (https://www.apple.com/ru/airport-time-capsule/)

M2M (machine-to-machine) – это решения, в основе которых заложено общение всяческих датчиков, сенсоров и прочего «железа» между собой на основе беспроводной связи

Примеры M2M:

- Банки и платежные системы (решения для соединения банкоматов, терминалов оплаты и других устройств с корпоративными IT-системами Клиента, прочие системы)
- Транспорт и логистика (решения для мониторинга автопарка, грузоперевозок, городского и муниципального транспорта и т.д.)
- Безопасность (решения по охране стационарных/подвижных объектов, видеонаблюдение и т.д.)
- Технологические АСУ естественных монополий и промышленных предприятий (энергетика, ЖКХ и т.п.)
- Потребительская электроника, вендинг и другие отрасли.
- Умные дома

Интернет вещей:

Устройства сбора информации (сенсоры, и тп) ->Cloud->Аналитическое ПО->Отображение и управление (мобильное приложение)

Параметры, определяющие устройства:

1. Дальность работы

2. Скорость работы
3. Объем передаваемых данных
4. Защищенность от зашумленности
5. Защищенность от злоумышленников
6. Энергопотребление

Используемые сети

1. Персональные – WPAN (Wireless personal area network) – до 10 м
2. Локальные – WLAN (Wireless Local Area Network) до 100 м
3. Масштабные – WMAN (Wireless Metropolitan Area Networks)

Передовой пример:
Semtech's LoRa (препод акцентировал на них внимание) – двунаправленная беспроводная технология, разработанная для разворачивания M2M инфраструктуры, предоставляющая недорогой вариант объединения различных устройств с батарей и мобильных устройств в единую сеть. Решение построено на собственном протоколе передачи данных LoRaWAN – использует свободные радиочастоты для предоставление не ресурсоемкого, широкоспектрального взаимодействия удаленных сенсоров (сборщиков данных) и шлюзами, подключенными к сети.
В России этим занимается компания LACE – они как раз предоставляют технологии LoRa
http://lace.io/ru/

Протоколы, используемые в интернете вещей.

Можно разделить на две группы:

1. (IEEE 802.15.4) – ZigBee, Wireless HART, Thread
2. (NoN - IEEE) – ANT/ANT+, Z-wave

Различные технологии:
LPWAN (*Low-power Wide-area Network*) - «энергоэффективная сеть дальнего радиуса действия») — беспроводная технология передачи небольших по объёму данных на дальние расстояния, разработанная для распределенных сетей телеметрии, межмашинного взаимодействия и интернета вещей. 10 -15 км
SigFox – работает в диапазоне 868MHz in Europe and 902MHz in the US
LTE – M от Nokia
NBLTE – M от Huawei