1. Обьяснить JDBC
2. Скачать Mysql
3. Сделать запросы по SQl

Maven обьяснить (*Перед тем как начать рассуждение на тему кросс-платформенных решений, нужно удостовериться, что мы четко понимаем разницу между кросс-платформенными фреймворками и нативной разработкой)*

МАТЧАСТЬ (КРОСПЛАТФОРМЕННЫЕ, НАТИВНЫЕ, ГИБРИДНЫЕ)

Перед тем как начать рассуждение на тему кросс-платформенных решений, нужно удостовериться, что мы четко понимаем разницу между кросс-платформенными фреймворками и нативной разработкой. Для простоты восприятия, мы будем проецировать процесс разработки кросс-платформенных приложений и сравнивать его с разработкой нативных приложений под iOS. Итак, какой же смысл скрыт за этими терминами.

НАТИВНЫЕ ПРИЛОЖЕНИЯ

Если говорить о разработке мобильных приложений под платформу Apple iOS, то нативным приложением будет называться решение, разработанное на языке программирования Objective-c или Swift (язык программирования Apple для разработки приложений под iOS и OS X, представленный на WWDC 2014), в рамках интегрированной среды разработки Xcode. С точки зрения пользователя, нативное приложение ничем не отличается от качественно выполненного кросс-плаформенного приложения, но все же стоит отметить, что нативные приложения под конкретную мобильную операционную систему, обычно, имеют понятный пользователю (нативный) интерфейс. Удобство заключается в том, что при разработке нативных приложений связка UI/UX специалиста и программиста наиболее эффективна: дизайнер точно знает какие UI подходы привычны пользователям конкретной операционной системы, например кнопка «Back» и нижний Tab Bar в iOS, а iOS разработчик точно знает как реализовать те или иные UI фишки именно для мобильных устройств производства Apple. Как результат, пользователь открывает нативное приложение и интуитивно взаимодействует с новым интерфейсом.

КРОСС-ПЛАТФОРМЕННЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

Важно понимать почему приложение называется кросс-платформенным. Дело в том, что, на самом деле, сам исполняющий файл, скомпилированный под одну из мобильных платформ, не может быть запущен в другой ОС. Т.е. если было разработано кросс-платформенное решение и скомпилировано под Android, это не означает, что можно взять файл с расширением «\*.apk» и запустить его на iPad’e. Идея кросс-платформенных решений не в удобстве для пользователя, а в оптимизации процессов разработки мобильного приложения. Следовательно, кросс-платформенное приложение - это решение разработанное таким образом, чтобы иметь возможность, с минимальными усилиями, скомпилировать исходный код для исполнения на нескольких мобильных платформах, но результатом каждой отдельной компиляции будут отдельные исполняемые файлы. Например, под iOS исполняемый файл имеет расширение - «\*.ipa», под Android - «\*.apk» и т.д.

ГИБРИДНЫЕ ПРИЛОЖЕНИЯ

Чаще всего это приложения, при разработке которых используются веб-технологии, а не разработка в нативной среде (пр. связка Xcode и Objective-c в случае разработки под iOS). В результате получается веб-приложение, которое запускается в «обертке» и подается не как веб-страница, а как отдельное приложение, требующее установки и имеющее отдельную иконку. Гибридные решения достаточно популярны среди разработчиков кросс-платформенных движков, обуславливается это тем, что с функцией веб-браузера может одинаково справиться практически любая мобильная ОС, а значит если приложение уже работает под какой-либо мобильной OS, то запустить ее на другой не составит труда. Конечно, есть свои нюансы, о которых мы поговорим в отдельном обзоре активно продвигаемого гибридного фреймворка, имя которому Phone Gap.

ПРЕДПОСЫЛКИ К ПОЯВЛЕНИЮ CROSS-PLATFORM РЕШЕНИЙ

Какие задачи призваны решить кросс-платформенные движки, почему кто-то решил, что в этом есть смысл? Если вы спросите у Java-разработчика под Android или Objective-c разработчика под iOS о том, что они думают по поводу кросс-платформенных фреймворков под мобильные платформы, то в большинстве случаев получите однозначную негативную характеристику и еще килограмм эпитетов :)

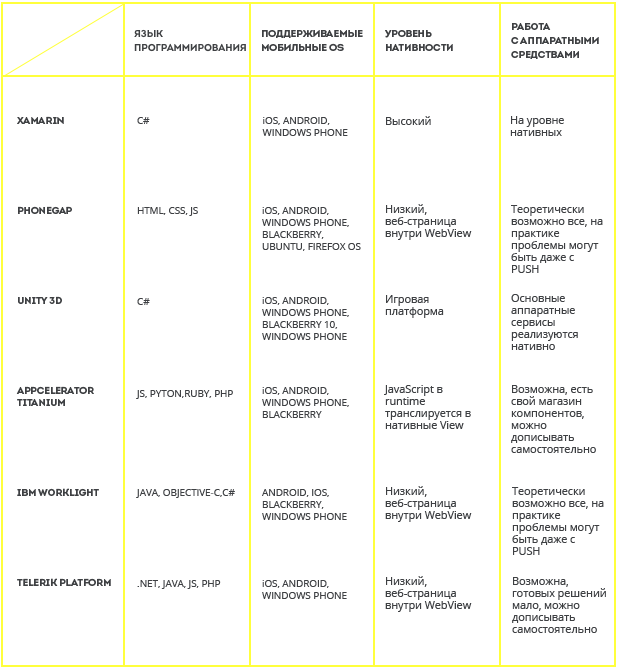
Основная причина такой реакции состоит в том, что хорошие разработчики, как и любой другой профессионал, очень трепетно относятся к самой концепции разработки, эффективности, производительности, у каждого из них свое видение того «что такое хорошо, а что такое плохо?», но обычно все крутится вокруг технологии которой они владеют, холят и лелеют. Важно понимать, что далеко не каждый разработчик может пожертвовать программистской романтикой и дать объективную оценку тому, что на самом деле выгодно для бизнеса в конкретный момент. В общем случилось так, что решает все, как обычно, бюджет, поэтому основной предпосылкой для появления кросс-платформенных решений становится именно польза для бизнеса, которую можно детализировать следующим образом:

* при разработке мобильного приложения под 2-3-х мобильных OS на кросс-платформенном фреймворке большая часть кода может быть написана одним специалистом. Т.е. отпадает необходимость в full-time разработчике под каждую отдельную OS;
* отсутствие на рынке рабочей силы узких специалистов под уже достаточно большое количество мобильных платформ;
* относительная простота в контексте контроля соответствий между версиями приложения под разными мобильными ОС.

Список этот довольно утопичен, но сейчас мы обсуждаем именно предпосылки, а они таковы :)

ЧТО ЕСТЬ И ЧТО НА САМОМ ДЕЛЕ КРУТО

Ниже список из 7 наиболее популярных платформ для разработки мульти-платформенных мобильных приложений. Кроме ссылок на сайты поставщиков решений мы также перечислим основные параметры, которые позволят поверхностно сравнить перечисленные платформы.



КРОСС-ПЛАТФОРМЕННЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

Важно понимать, что существует множество причин, по которым один разработчик выберет одну платформу, а другой другую. Например, рубист не посмотрит в строну Xamarin, потому что у него, скорее всего, отсутствует опыт разработки на C#. Но если Xamarin окажется в руках опытного C# разработчика, то в результате клиент получит приложения под Android и iOS, которые ничем не уступают нативным.

С другой стороны, несмотря на то, что в команде есть 2-3 отличных C# девелопера, Xamarin может оказаться дорогим в реализации, уступив дешевому PhoneGap, если проект не требует никакой особенной работы с графикой и аппаратными средствами устройств, а планируется лишь как MVP прототип для проверки бизнес модели и должен быть реализован в рамках крайне ограниченного бюджета.

Еще одним достаточно существенным параметром является цена за доступ к среде разработки на конкретном фреймворке. Если PhoneGap полностью бесплатный инструмент, то Vеrivo хочет $60,000/год, что вызовет недоумение у стартаперов, да и не только у них :)

Connect c mysql Database

package data;  
  
import com.mysql.cj.conf.ConnectionUrl;  
  
import java.sql.Connection;  
  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
  
public class Main {  
  
  
  
 public static void main(String[] args) throws ClassNotFoundException, SQLException {  
 String url = "jdbc:mysql://localhost:3306/test?serverTimezone=UTC&useSSL=false";  
 String username ="root";  
 String password = "muhit111";  
  
  
 Class.*forName*("com.mysql.jdbc.Driver");  
 try (Connection connection = DriverManager.*getConnection*(url,username,password) ){  
 System.*out*.println("We connected");  
 }  
 }  
}