Разработка кросс-платформенных мобильных приложений: проблемы и решения

Авторы доклада: Александр Коршак, Максим Лейкин,
департамент мобильных приложений ООО «МЕРА НН»

Нижний Новгород, 2013

1. Наш опыт
2. Типы мобильных приложений
3. Тенденции кросс-платформенности
4. Кросс-платформенность в нативных приложениях
   - Схемы реализации
   - Преимущества
   - Специфика
   - Проблемы и решения
5. Гибридные мобильные приложения
   - Среды разработки
   - Архитектура
   - Тенденции рынка
   - Преимущества и специфика
   - Проблемы и решения

Департамент мобильных приложений Мера-НН:

50+ mobile projects, 400+ man-years

- ✓ **Android, iOS, Windows Phone 8 / RT**
- ✓ **BlackBerry 10 / QNX, J2ME**
- ✓ **HTML5 / JS / CSS**
- ✓ **Firefox OS**
- ✓ **PhoneGap/Cordova API, Marmalade**

Ключевые кросс-платформенные проекты:

- ✓ **разработка и сопровождение мобильного навигационного приложения для платформ Android/iOS/BB 7/BB10/Widows Phone/BREW**
- ✓ **разработка новой функциональности в существующем гибридном мобильном приложении, предназначенном для управления smart home устройствами**

Out of scope of the paper

Apps developed exclusively for a specific mobile platform that can leverage all device capabilities.

Apps that wrap a mobile web interface inside a native container

Apps implemented with HTML5 and JavaScript that operate entirely inside a mobile browser.

Mobile App Technology Stacks
Your App
Operating System & Device
Native App
Your App
Embedded Web Views
Native UI Views
Native Hardware Access
Native Wrapper App
Operating System & Device
"Hybrid" App
Your App
Web Browser
Operating System & Device
Web App

C - Based

- Brew
- iPhone
- BB10
- Windows Phone 8
- Android (NDK)

JVM - Based

- RIM Blackberry
- Android
- J2ME

Brew
Blackberry
Android
Windows Phone 8
Client UI
Interfaces
Business Logic
Cross-platform SDK
Brew
Blackberry
Android
Windows Phone 8
PALIMPL

Platform Abstraction Layer (C-Based + POSIX)	
Cross-Platform PAL components	Platform specific PALIMPL components
Database	Archiving
Date/Time	Database
Network	Graphics and rendering
Synchronization	File System
	Radio information
	Logging

MediaPlayer
IPAL
«interface»
PALInstance
AudioRecorder
«interface»
AudioBuffer
AudioFormat
AudioException
PalImpl
AudioRecorderImpl
Business
Logic
Common
PAL
PALIMPL

```
class MediaPlayer() {
   AudioRecorder mRecorder;
   MediaPlayer() {
      IPAL pal = PALInstance.getPal();
      mRecorder = pal.getAudioRecorder();
   }
   …
}
```

```
class PalImpl implements IPAL {
   private AudioRecorder mRecorder;
   PalImpl() {
      mRecorder = new AudioRecorderImpl();
   }
   public AudioRecorder getAudioRecorder() {
      return mRecorder;
   }
   …
}
```

```
class AudioRecorderImpl implements AudioRecorder
{
   private AudioBuffer mBuffer;
   private AudioFormat mFormat;
   private AudioException mException;
   …
   AudioRecorderImpl() { … }
}
```

- **Skill Set**
- **Объем кода**

- **Время разработки**
- **Процесс интеграции**
- **Bug rate & maintenance**

JVM-Based	C-Based
Организация процесса	
Билд-скрипты	
Code Review	
Unit Tests	
Ограничения	
Максимально допустимы API не выше 4 версии	Использование STL в публичных методах
Необходимо изобретать «велосипед»	Обработка исключений
	Использование системных библиотек и классов
	Работа с UI потоком
	Использование Timer (BB10)
	Обратная совместимость Win-phone платформы
	Специфика Win8 Network API
	Rendering. OpenGL context using

Проблемы
Неудобство отладки native-кода на Android
Сложности с применением STL в публичных методах
Особенности использования OpenGL на разных платформах
Ограничения в использовании системных библиотек и классов
Особенности работы с UI-потоком
Трудности процесса интеграции

Решение:

- **NDK-STACK**

  adb logcat | $NDK/ndk-stack -sym $PROJECT_PATH/obj/local/armeabi

До	I/DEBUG (31): *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** I/DEBUG (31): Build fingerprint: 'generic/google_sdk/generic/:2.2/FRF91/43546:eng/test-keys‘ I/DEBUG (31): pid: 351, tid: 351 %gt;%gt;%gt; /data/local/ndk-tests/crasher <<< I/DEBUG (31): signal 11 (SIGSEGV), fault addr 0d9f00d8 I/DEBUG (31): r0 0000af88 r1 0000a008 r2 baadf00d r3 0d9f00d8
После	********** Crash dump: ********** Build fingerprint: 'generic/google_sdk/generic/:2.2/FRF91/43546:eng/test-keys‘ pid: 351, tid: 351 >>> /data/local/ndk-tests/crasher <<< signal 11 (SIGSEGV), fault addr 0d9f00d8 Stack frame #00 pc 0000841e /data/local/ndk-tests/crasher : Routine zoo in /tmp/foo/crasher/jni/zoo.c:13

- **NDK-Plugin**

  Note: очень «капризный»

- **CDT-Plugin (Eclipse) + CYGWIN (Windows) + NDK-GDB**

  1) **ndk-build NDK_DEBUG=1**
  2) **add C/C++ nature into project**
  3) **breakpoints**
  4) **ndk-gdb --force –verbose**
  5) **welcome**

Пример:

```
vector <int>::const_reverse_iterator v1_rIter;
v1.push_back( 1 ); v1.push_back( 2 );
v1_rIter = v1.rbegin( );
```

const_reverse_iterator – не скомпилируется на Android и Brew

P.S.:

В Brew реализовано около 5 базовых классов STL (Vector, Map…). Названия методов могут меняться в зависимости от платформы.

Возможные решения:

- **Избегать использование STL в Cross-Platform компонентах**
- **Использование собственной реализации объектов STL**
- **Ручная проверка работоспособности билд-скриптами**
- **Повышенные требования к экспертизе команды в целом + инспекция кода**

```
struct PAL_Vector
{
   size_t itemsize;
   int size;
   int allocated;
   void *items;
};

struct PAL_Vector* PAL_VectorAlloc(size_t itemsize);
struct PAL_Vector* PAL_VectorAllocPlace(size_t itemsize, void* p);
int PAL_VectorAppend(struct PAL_Vector *vector, const void *item);
void PAL_VectorDealloc(struct PAL_Vector *vector);
void PAL_VectorDeallocPlace(struct PAL_Vector *vector);
void PAL_VectorGet(struct PAL_Vector *vector, int index, void *item);
void* PAL_VectorGetPtr(struct PAL_Vector *vector, int index);
void PAL_VectorRemove(struct PAL_Vector *vector, int index);
int PAL_VectorRemoveItem(struct PAL_Vector *vector, void *item);
```

Особенности использования OpenGL на разных платформах

Примеры:

- **Инициализация Surface объекта специфична на каждой платформе**
- **Возможны различия в жизненном цикле OpenGL контекста**
- **Система координат платформы**

- **Различия в публичных API и константах (Пример: GL_UNSIGNED_INT, GL_DEPTH_COMPONENT16, glGenFramebuffers отсутствуют на Android, необходимо использовать 0x1405, GL_DEPTH_COMPONENT16_OES, glGenRenderbuffersOES)**
- **OpenGL официально не поддерживается в Windows Phone 8**

Возможное решение:

- **Реализация собственной “Render engine”.**

Render Instance
IRenderer <interface>
Render Factory
GLES2 Renderer
DirectX 9 Renderer

Ограничения в использовании системных библиотек и классов

Пример:

```
const char *url = "http://www.meranetworks.com/";
size_t slen = strlen (url);
char strdest[8];
strlcpy(strdest,pObjUrl,slen+1);
```

strlcpy – не скомпилируется на Brew, Android, Symbian

Возможные решения:

- **Использование собственной реализации функций из stdlib.h и math.h**
- **Переопределение нестандартных типов из stdint.h (пр. uint32)**

Примеры:

- **Windows Phone 8 – с callback`ом можно вернуть объект только если он инициализирован в UI потоке**
- **Blackberry 10 – при отказе от использования QT, callback из UI потока нельзя использовать**

Решение:

- **Использование собственной реализации UI потока и исполняемых очередей**

Проблема:

- **Клиент-серверное приложение**
- **При загрузке данных из сети изредка происходит потеря части пакетов (Map Tiles)**
- **Примерно 50000 строк кода для анализа, 5 поддерживаемых платформ**
- **Область поиска проблемы – PAL, Business Logic, Server, Client**

Решение и спасение:

- **Unit-тесты для каждой из платформ + бизнес-логики**

Результат:

- **Unit-тесты для Windows Phone реализации Network-интерфейсов помогли вычислить неправильный порядок State’ов соединения и найти проблемный компонент.**

PhoneGap	PhoneGap www.phonegap.com
titanium	Appcelerator Titanium www.appcelerator.com/titanium/
Sencha	Sencha Touch www.sencha.com/products/touch
Marmalade	Marmalade www.madewithmarmalade.com

Cross-platform Tools

Source: Developer Economics 2013 | www.DeveloperEconomics.com | January 2013
Licensed under Creative Commons Attribution 3.0 License

на примере PhoneGap:

По оценкам Gartner Group (http://androidcommunity.com/gartner-by-2016-more-than-half-of-all-mobile-apps-will-be-hybrid-20130204/) к 2016 году более половины мобильных приложений станут гибридными

Features	Native	Hybrid	Web
Development Language	Native only	Native and Web / Web only	Web only
Code Portability	None	High	High
Device Specific Features	High	Moderate	Low
Access to Native APIs	High	Moderate	None
Advanced Graphics	High	Moderate	Moderate
UI / UX	High	Moderate	Low
Upgrade Flexibility	Low (always through appstore)	Moderate (generally through appstore)	High
Installation Experience	High (from appstore)	High (from appstore)	Low (via mobile browser)

Проблемы
Создание UI, учет UX для разных платформ
Сложность выполнения фоновых потоков
Снижение производительности по сравнению с нативным кодом.

Предположение: кросс-платформенный фреймворк = кросс-платформенный UI?

1. **Необходимо делать отдельные UI specs**
2. **Необходимо учитывать UX каждой платформы**
3. **Необходимо тестировать UI на разных устройствах**

Большое кол-во приложений использует фоновые потоки для выполнения части функциональности. Поскольку PhoneGap построен на JavaScript, который не является многопоточным, перевод таких приложений на гибридную архитектуру представляет собой проблему.

Решение: разработка специального плагина на Java:

```
public class BackgroundService extends Plugin {
private static final String TAG = "BackgroundService";
private static final String CALL_SERVICE_ACTION = "callService";
public PluginResult execute(String action, JSONArray args, String callbackId)
 {  PluginResult result = null;
    if (CALL_SERVICE_ACTION.equals(action)) {
        startService(new Intent(ctx, MyService.class));
     }
    else {
        result = new PluginResult(Status.INVALID_ACTION);
    }
    return result;
  }
}
```

И вызов его через HTML/JS код:

```
function BackgroundService() {
};
BackgroundService.prototype.callService = function(successCallback, failCallback)
{  return PhoneGap.exec(successCallback, failCallback, "BackgroundService", "callService", [ null ]);
};
PhoneGap.addConstructor(function() {
   PhoneGap.addPlugin("BackgroundService", new BackgroundService());
});
```

```
function callServiceFunction() {
window.plugins.BackgroundService.callService('callService',
       callServiceSuccessCallBack, callServiceFailCallBack);

}
function callServiceSuccessCallBack(e) {
alert("Success");
}

function callServiceFailCallBack(f) {
alert("Failure");
}
```

Внимательно прослушать:
http://coenraets.org/blog/2013/05/top-10-performance-techniques-for-phonegap-applications
/ и применять со

Top 10 Performance Techniques for PhoneGap Applications

by **Christophe Coenraets** on May 13, 2013 in **PhoneGap**

Here is the video of my PhoneGap performance session at Adobe MAX 2013:

Q&A