СРАВНЕНИЕ ПОДХОДОВ К НАПИСАНИЮ ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Строенко Ю.А.

Сиротко С.И. – канд. техн. наук, доцент

В наше время все сильнее развивается рынок мобильных устройств, и, следовательно, все более востребованы различного рода приложения под мобильные телефоны, планшеты.

Это естественно, что в данной отрасли существует серьезная конкуренция между производителями мобильных аппаратов и программного обеспечения для них. На данный момент самыми крупными мобильными платформами являются: iOS от Apple, Android от Google, Windows Phone от Microsoft и другие.

Для каждой из этих платформ существуют собственные языки программирования, интегрированные среды разработки и множество других вспомогательных технологий. Конечно же, ни о какой совместимости нативных приложений с различными платформами речь не идет.

Однако разработчики приложений практически всегда имеют цель выпустить продукт, который был бы доступен владельцам смартфонов и планшетных компьютеров различных производителей. Логичный выход из этой ситуации – разрабатывать отдельное приложение для каждой целевой платформы.

Одним из альтернативных подходов к решению данной проблемы является разработка веб-приложений, выполняемых в браузере. Дело в том, что практически все браузеры интерпретируют HTML-разметку и язык JavaScript одинаково, следовательно, приложение будет выглядеть и работать одинаково на различных устройствах. Однако такой подход имеет большой ряд недостатков. Основным из которых является отсутствие доступа к низкоуровневым API устройства. А также некоторые другие ограничения, такие как невозможность запуска Flash-приложений в браузерах операционной системы iOS.

В последние несколько лет было создано несколько сложных платформ разработки и развертывания кроссплатформенных приложений, использующих кардинально другой подход. Они также позволяют создавать универсальные приложения для мобильных устройств, используя JavaScript, HTML5 и CSS3, однако ключевое отличие состоит в том, что готовое приложение компилируется в виде установочных пакетов для каждой мобильной операционной системы. Эти технологии называются гибридными, потому что они не являются нативными(вся разметка пользовательского интерфейса выполняется с помощью веб-представлений, вместо “родной” UI-платформы данной операционной системы), и в то же время не являются веб-приложениями(они имеют доступ к некоторым низкоуровневым API).

Наиболее известными гибридными платформами являются PhoneGap от Adobe и Titanium от Appcelerator.

Последний подход имеет свои преимущества и недостатки. Основным достоинством является уменьшение времени разработки, что приводит к удешевлению конечного продукта. Это простая логика, которая заключается в том, что разработчикам необходимо создать один продукт, чтобы он работал на, скажем, трех платформах, вместо того, чтобы создавать три продукта. PhoneGap и Titanium также предоставляют быстрый и легкий доступ к плагинам, которые могут быть использованы для работы с API платформы. Например, они имеют средства работы с GPS. Вместо того, чтобы писать уникальный код для iOS и Android, разработчик может использовать плагин, который “сделает“ все за него. Также, благодаря тому, что разработка ведется на JavaScript и HTML5, на неё могут безболезненно переключиться веб-разработчики.

К недостаткам данных технологий относится тот факт, что их создателям очень сложно их поддерживать. Если выходит какая-нибудь новая функциональность от производителей одной из операционных систем, она не сразу поддерживается гибридными платформами. Это неизбежно – приходилось бы делать огромное количество обновлений, чтобы сразу же включать все нововведения любой из поддерживаемых операционных систем. Также, многие низкоуровневые API остаются недоступными извне, а только лишь с использованием приватных фреймворков и языков программирования. Например, с помощью PhoneGap невозможна работа с OpenGL для iOS.

Однако главным недостатком данного подхода является более низкая производительность, в отличие от нативной разработки. Гибридное приложение как правило уступает в производительности при работе с низкоуровневыми API устройства, будь то акселерометр, камера, GPS, календарь, микрофон и другие.

В качестве средств анализа и сравнения производительности нативных и гибридных мобильных приложений можно использовать:

1. Сравнение “на глаз”. Самый простой способ, который может показать примерную разницу в производительности. Для этого необходимо 2 приложения, выполняющих одинаковые операции, однако разработанные при помощи различных технологий: первое – при помощи гибридных, другое – при помощи естественной среды разработки. На двух одинаковых устройствах запустить выполнение данных операций и сравнить результат.
2. Сравнение при помощи уже существующих инструментов. Например, для устройств Apple существует ряд инструментов в среде Xcode, которые предназначены именно для тестирования работы прило-

жения. Кроме времени выполнения операции, можно оценить загрузку ЦП и количество используемой памяти при работе приложения.

3. Разработка нативного приложения, которое будет содержать в себе элементы гибридного приложения. При этом различные части приложения будут выполнять одинаковые функции, а приложение будет фиксировать время выполнения для каждого из вариантов. Как результат, можно вывести сводную сравнительную информацию о времени выполнения аналогичных операций.

В качестве вывода следует отметить, что при выборе способа разработки приложений необходимо четко понимать, какие цели оно будет преследовать и какие функции выполнять. Если это достаточно простое приложение, которое должно работать на различных платформах и не требующее доступа к низкоуровневым функциям устройства, следует выбирать одну из гибридных технологий. Если же это сложный продукт, использующий множество аппаратных ресурсов, то лучшим вариантом будет разработка нескольких вариантов нативных приложений.

Список использованных источников:

1. А.Хиллегасс, Программирование под iOS для професисоналов, 608 стр. (2013 г.);
2. Lee Barney, Developing Hybrid Applications for the iPhone, 183 стр. (2009 г.).

МУЛЬТИПЛАТФОРМЕННЫЙ СЕРВИС ЛОГИРОВАНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Шабанец Я. Р.

Волосевич А. А. – канд. физ.-мат. наук, доцент

Грамотно спроектированная архитектура современных программных продуктов включает в себя модули обработки ошибок и ведения журнала событий, возникающих в ходе работы приложения. Возникновение ошибок в программе во многих случаях обусловлено некорректными данными, переданными приложению, либо некорректной последовательностью действий пользователя при работе с программой. Для обнаружения и устранения проблем, обнаруженных в системе, разработчики используют записи из журналов ошибок и сообщений приложения для восстановления контекста выполнения, приведшему к возникновению проблемы.

Мультиплатформенный сервис логирования предназначен для ведения журнала сообщений приложения, их хранения и фильтрации, предоставления статистики в режиме реального времени. Сервис является готовым решением, которое можно применять в существующих или проектируемых системах без необходимости разработки собственного решения, уменьшая время, необходимое на разработку и затраты на создание программных продуктов.

Проект состоит из нескольких частей: сервис приема сообщений, сервис поставки сообщений, компонент для высокоэффективной работы с хранилищем сообщений, библиотеки классов для различных платформ, содержащие в себе модули для работы с сервисами и веб интерфейс сервиса.

Классы для работы с сервисом предоставляют разработчикам несколько перегруженных методов: Debug, Info, Warn, Error, Fatal для отправки сообщений на сервис и возможность подписывать собственные обработчики сообщений, получаемых сервисом из других уровней приложения.

Веб интерфейс сервиса предоставляет доступ к архиву сообщений приложения с возможностью экспорта, фильтрации и сортировки данных.

Сервис реализован с использованием следующих технологий платформы .NET: ASP.NET MVC4 для реализации веб интерфейса, ASP.NET Web API для реализации сервисов приема и поставки сообщений.

Для хранения данных, в целях повышения производительности, было выбрано документо-ориентированное сетевое хранилище данных типа «ключ-значение» с открытым исходным кодом Redis. Redis хранит базу данных в оперативной памяти, снабжена механизмами снимков и журналирования для обеспечения постоянного хранения.

Отличительной чертой реализации логирования является доступ и синхронизация сообщений всех уровней приложения, которые могут быть разделены физически и написаны с использованием различных технологий. Также ведение журнала событий потребляет ресурсы приложения, таких как дисковое пространство, процессорное время и оперативная память. Использование сервиса позволяет избежать подобных затрат на серверах разрабатываемой системы и улучшить ее нагрузочные характеристики.

Разработанное решение может применяться для ведения журнала сообщений приложений и может использоваться в компаниях, занимающихся разработкой программного обеспечения, для диагностики создаваемых или сопровождаемых программных продуктов.

Список использованных источников:

1. Tiago Macedo, Fred Oliveira Redis Cookbook / Tiago Macedo, Fred Oliveira – O'Reilly Media 2011. – 78 p.
2. Jamie Kurtz ASP.NET MVC 4 and the Web API / Jamie Kurtz – Apress, 2013. – 152 p.