Информационные системы на основе push-уведомлений

Павлов А.Д., Намиот Д.Е.

***Аннотация*—Настоящая работа посвящена задаче построения информационных моделей на основе push–уведомлений для мобильных клиентов. Целью работы является анализ возможности построения информационных систем на основе push-уведомлений без программирования. Конечная цель – это построение некоторого простого аналога специализированной CMS (Content Management System), ориентированной на распространение информации с помощью серверных уведомлений. В статье представлена базовая реализация модели регистрации подписчиков и рассылки push-уведомлений.**

***Ключевые слова*— push, уведомления, мобильный клиент, CMS.**

I. Введение

В данной статье представлены результаты квалификационной работы, выполненной в лаборатории Открытых Информационных Технологий факультета ВМК МГУ имени М.В. Ломоносова. Работа была посвящена исследованию возможности создания информационных систем на основе push-уведомлений. В первой части опубликованного исследования мы остановились непосредственно на поддержке push-уведомлений различными сервисами [1]. Данная публикация посвящена собственно информационным системам.

Краткое введение в проблему. Речь идет об очевидном тренде, в котором мобильные абоненты отказываются от традиционных для телекоммуникационных систем уведомлений (SMS, MMS) в пользу сообщений от разного рода мессенджеров [2]. Очевидно, что такая замена должна коснуться и использования уведомлений в разного рода приложениях (информационных сервисах). SMS – это традиционный канал рассылки уведомлений. Простой в реализации и удобный для пользователей. Одна из возможных замен, например, это сервисы в социальных сетях и мессенджерах (боты) [3][4]. Другая возможность, которая и является темой данного исследования – это push-уведомления.

Статья получена 10 июля 2014.

А.Д. Павлов – выпускник программы РКТ факультета ВМК МГУ имени М.В.Ломоносова. (email: a.d.pavlov@yandex.ru)

Д.Е. Намиот – старший научный сотрудник лаборатории ОИТ, факультета ВМК МГУ имени М.В.Ломоносова. (email: dnamiot@gmail.com)

Push-уведомления (server push) – это широко используемый в мобильных приложениях механизм. Он поддерживается практически всеми производителями мобильных операционных систем (ОС). Именно мобильные ОС отвечают за доставку этих сообщений. И адресатом сообщений (подписчиком), в отличие от SMS, является не мобильный абонент, а конкретное приложение.

Благодаря поддержке производителей мобильных ОС (а такой механизм с примерно одинаковой функциональностью присутствует во всех современных операционных системах), push-уведомления широко используются в мобильных разработках. Например, в работах, выполненных в лаборатории ОИТ [5][6]. Технология push (server push) описывает один из способов доставки контента мобильным пользователям посредством сети Интернет. Мобильный клиент (subscriber) при помощи установленного приложения подписывается на рассылку push-уведомлений, а сервер публикаций (publisher), по наступлению определенного события, осуществляет рассылку подписавшимся клиентам. Таким образом, адресатом сообщения всегда является некоторое приложение. Важно отметить, что push–уведомления позволяют серверам рассылки уведомлять пользователей о наступлениях событий, даже если приложение в данный конкретный момент неактивно. Достаточно, чтобы приложение было установлено, и действовала подписка.

Отправка сообщения (через посредника в лице сервиса мобильной ОС) организуется некоторым внешним приложением. Приложение может, например, реагировать на некоторое внешнее событие (ввод нового сообщения администратором). Возможна также организация некоторой гео-решетки (точнее – ее аналога на идеях сетевой близости [7]). Например, с помощью пассивного Wi-Fi мониторинга [5], можно определять мобильных абонентов поблизости от некоторой точки доступа Wi-Fi, а затем рассылать сообщения только тем подписчикам, которые в данный момент находятся рядом [6]. Аналогичная схема может быть реализована на основе Bluetooth. При этом возможно как использование Bluetooth Low Energy [8], так и Core Bluetooth [9].

В первой части исследования были рассмотрены системы для поддержки отправки сообщений [1]. Из новых работ в этой области можно отметить еще Aerogear [10]. Эта система может быть весьма привлекательна для Java разработчиков, поскольку базируется на сервере JBoss. В данной же работе мы

хотим рассмотреть уже собственно информационные системы. При этом задача состоит в создании таких сервисов, которые позволят использовать рассылку уведомлений без программирования (или с минимальным объемом такового). В идеале, это должна быть некоторая content management system (CMS) a-la Drupal [11], например. Пользователи должны иметь возможность организовать рассылку уведомлений (по крайней мере, в некоторой базовой конфигурации), непосредственно после установки. Очевидно, что это включает в себя не только какую-то серверную установку, но и стандартный мобильный клиент.

Drupal был приведен в качестве примера не случайно. Эта оболочка широко используется, например, в подразделениях Московского Государственного Университета. Очевидно, что сайты кафедр (лабораторий) не ограничиваются только публикацией статических файлов (курсов лекций и т.п.). Есть довольно много разовой информации (изменения в расписании, необходимость предоставления документов и т.п.), которая должна как-то сообщаться студентам и сотрудникам. Это типичные примеры для возможных рассылок.

Еще больше данных подобного рода будет у предприятий обслуживания и торговли (HoReCa). Адресатом подобного рода сервиса могут являться как конечные пользователи, так и другие предприятия. Очевидно, что число возможных потребителей больше, чем доступное число разработчиков. Нам представляется, что такой сервис будет явно востребован и может иметь явную коммерческую направленность.

Работа выполнялась в рамках исследований, проводимых в лаборатории ОИТ [12].

II. Модели информационных систем

В дальнейшем рассмотрении мы предполагаем, что имеется (выбрана) некоторая служба уведомлений (PSP – Push Service Provider), из описанных в первой части исследования [1].

A. Модель регистрации подписчиков и посылки native push-уведомлений

Схема работы проиллюстрирована на рисунке 1.

Регистрация подписчиков

1. Подписчик (Subscriber) регистрируется на сервере собственной службы уведомлений (PSP).
2. В ответ подписчик получает регистрационный номер (для GCM – registrationID, для APNS – deviceToken).
3. Подписчик передает регистрационный номер серверу публикаций (Publication server). Также он может указать тему (topic) подписки или набор тегов (tags).
4. Сервер публикаций сохраняет регистрационный номер подписчика в базе данных.

Посылка native push–уведомлений

1. Публикатор размещает уведомление (оно может относиться к какой-либо теме, либо к сегменту пользователей, либо быть индивидуальным) либо с использованием API (удаленно), либо с использованием web – интерфейса (на стороне сервера публикаций).
2. Сервер публикаций выполняет задачу рассылки уведомлений соответствующим PSP с использованием регистрационных номеров подписчиков.
3. PSP отслеживает состояние подписчика (в сети или нет) и осуществляет дальнейшую доставку подписчику (мобильному устройству) с соответствующим регистрационным номером.

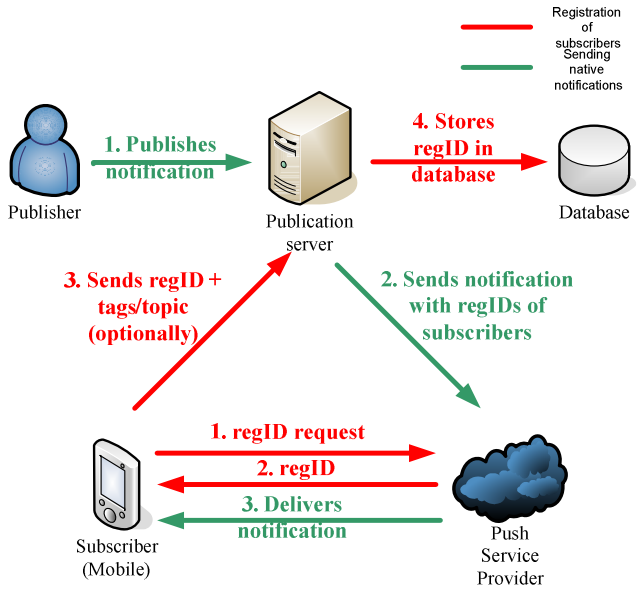

Рисунок 1. Модель регистрации подписчиков и посылки native push-уведомлений

B. Модель посылки rich push-уведомлений

Напомним, что rich-уведомления позволяют рассылать не только текст, но и мультимедийный контент [1]. Этот контент должен где-то храниться. Как правило, это реализуется в виде некоторой специально создаваемой HTML страницы (страниц). Схема работы проиллюстрирована на рисунке 2.

1.Публикатор создает rich push-уведомление (например, в редакторе сервера публикаций), публикует его подписчикам какой-либо темы, сегменту (группе) пользователей, либо посылает индивидуально.

2.Сервер публикаций извлекает из базы данных необходимые регистрационные номера подписчиков.

3.Сервер публикаций выполняет рассылку уведомлений соответствующим PSP с использованием регистрационных номеров с учетом тем/тегов. В теле сообщения передается rich push ID – сформированный сервером публикаций.

4.PSP отслеживает состояние подписчиков (в сети или нет) и осуществляет дальнейшую доставку.

5.Подписчик передает rich push ID серверу публикаций.

6. Сервер публикаций возвращает HTML страничку.

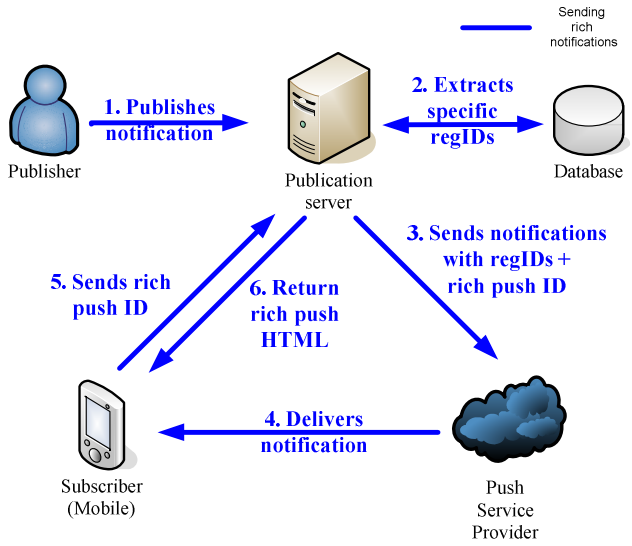

Рисунок 2. Посылка rich push-уведомлений

III. Виды рассылок push-уведомлений

В настоящем разделе рассматриваются виды рассылок.

A. *Индивидуальная рассылка*

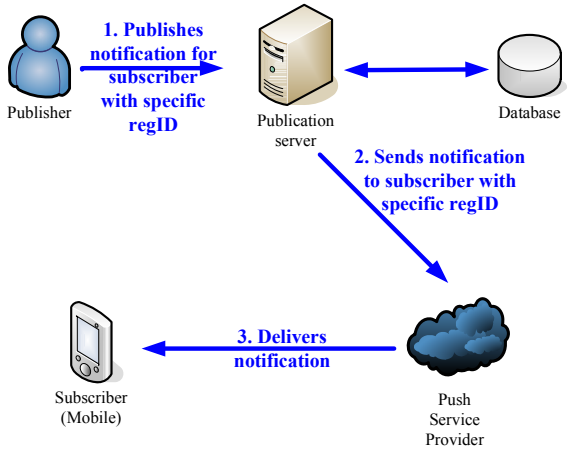

Рисунок 3. Индивидуальная рассылка

1.Публикатор размещает уведомление для подписчика с уникальным регистрационным номером.

2.Сервер публикаций осуществляет отправку уведомления с указанным *regID* соответствующему PSP.

3. PSP отслеживает состояние подписчиков (в сети или нет) и осуществляет дальнейшую доставку.

B. *Широковещательная рассылка*

1.Публикатор размещает уведомление в определенное приложение (application).

2.Сервер публикаций извлекает регистрационные номера подписчиков.

3. Сервер публикаций осуществляет рассылку уведомления с указанными *regID* соответствующим PSP.

4. PSP отслеживает состояние подписчиков (находится подписчик в сети или нет) и осуществляет дальнейшую доставку.

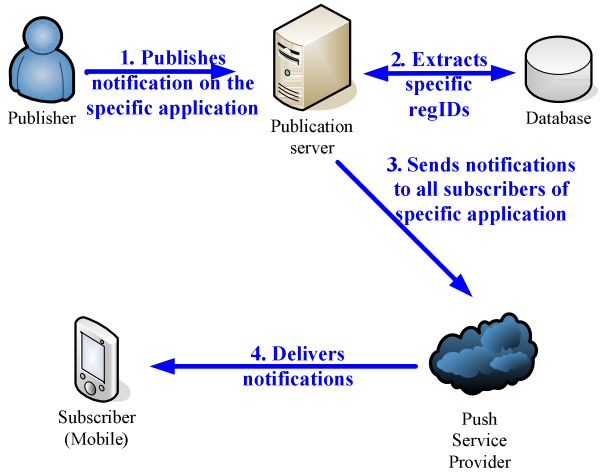

Рисунок 4. Широковещательная рассылка

C. *Сегментная рассылка*

Здесь имеется в виду сегментация подписчиков. Рассылка сообщений подписчикам, которые относятся к некоторой группе, характеристики (описание) которых удовлетворяет некоторому условию и т.д.

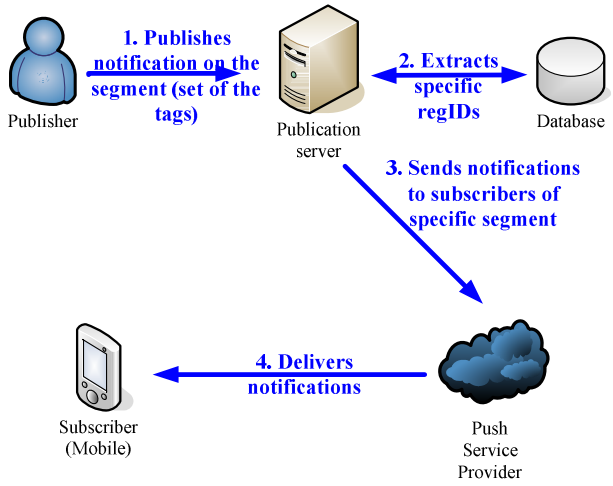

Рисунок 5. Модель сегментной рассылки

1.Публикатор (издатель) размещает уведомление в определенный сегмент (набор тегов с логическими операциями, возможно указание диапазонов для значений тегов).

2. Сервер публикаций извлекает регистрационные номера подписчиков, удовлетворяющие определенным требованиям.

3. Сервер публикаций осуществляет рассылку уведомления с указанными *regID* соответствующим PSP.

4. PSP отслеживает состояние подписчиков (в сети или нет) и осуществляет дальнейшую доставку.

D. Гео-рассылка push-уведомлений

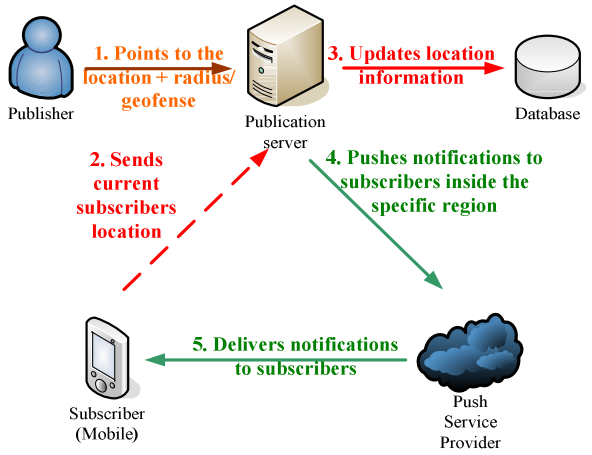

Рисунок 6. Гео-рассылки

1.Публикатор указывает серверу публикаций координаты точек с радиусами действия, координаты зон.

2.Подписчик после регистрации с определенной периодичностью посылает серверу публикаций свои координаты.

3.Сервер публикаций обновляет координаты подписчиков в базе данных.

4.Сервер публикаций извлекает регистрационные номера подписчиков, находящихся в определенном регионе и осуществляет рассылку уведомлений с указанными *regID* соответствующим PSP.

5.PSP отслеживают состояния подписчиков и осуществляют дальнейшую доставку.

На позиции 2 в этой последовательности шагов стоит остановиться подробнее. В классической модели – да, все обстоит именно так, как и указано. Подписчик уведомляет о своем местоположении. Если подписчик – мобильный пользователь, то это означает наличие у него некоторого приложения (запущенного, при этом), которое отслеживает его перемещения. Это может оказаться не очень приемлемым для многих пользователей. Совсем другая картина получится, если мобильное устройство (прозрачно для пользователя) будет опрашиваться о текущей позиции. Вот в работе [7] можно почитать про OMA SUPL в этой связи.

Здесь же необходимо остановиться и на возможной замене гео-координат информацией о сетевой близости. Иными словами, речь идет о позиционировании мобильного устройства (и мобильного абонента, соответственно) относительно сетевых узлов беспроводной сети (точек доступа Wi-Fi, в первую очередь) [13]. Дело в том, что такого рода мониторинг может быть пассивным, то есть не требовать от мобильного абонента установки и запуска каких-либо приложений. Соответственно, сторонний сервис будет получать информацию о местоположении мобильного абонента. И эту информацию уже нужно будет сопоставлять с базой данных подписчиков.

В работе [6] это описано подробно. Сервис мониторинга определяет (идентифицирует) мобильных абонентов по MAC-адресам их устройств. Соответственно, информация о подписке должна включать в себя и MAC-адрес подписчика (тогда можно будет определить, является ли подписчиком обнаруженный мобильный пользователь).

E. Рассылка на основании информации от тегов

Это уже в чистом виде использование network proximity (сетевой близости). В качестве таковых тегов могут выступать Bluetooth метки (iBeacon) [14], или же мобильные телефоны (другие устройства) с Core Bluetooth в discoverable mode. Этот случай отличается от гео-рассылок, поскольку сам телефон (устройство подписчика) невозможно зарегистрировать (обнаружить) сторонним сервисом. Активным элементом является тег. Устройство подписчика (его мобильный телефон) может обнаружить тег и уведомить об этом сервер подписки. А уже там, к факту присутствия около некоторого тега может привязана рассылка.

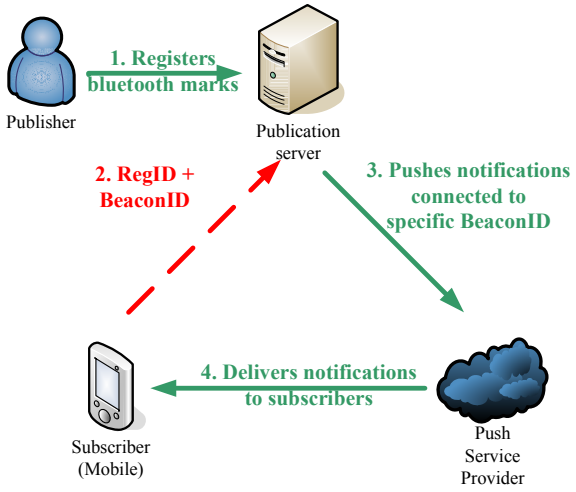

Рисунок 7. Рассылка на основе меток Bluetooth

1. Публикатор регистрирует на сервере публикаций Bluetooth–метки.

2. Подписчик обнаруживает Bluetooth–метку и отсылает серверу публикаций свой регистрационный номер и идентификатор метки (*BeaconID*).

3. Сервер публикаций определяет содержимое уведомления по присланному идентификатору метки. Сервер публикаций осуществляет посылку уведомления с указанным *regID* соответствующему PSP.

4. PSP отслеживает состояния подписчика и осуществляют дальнейшую доставку.

IV. Реализация модели регистрации и посылки native push-уведомлений

Модель регистрации подписчиков и посылки native push-уведомлений является базовой для любого сервиса push-уведомлений. Критериями выбора сервиса выберем простоту, удобство и скорость организации на его основе публикации и доставки push-уведомлений мобильным клиентам под Android.

Систему рассылки, реализующую модель регистрации подписчиков и посылки native push-уведомлений с минимальными затратами, будем называть элементарной.

Рассмотрим организацию элементарной системы рассылки на основе сервисов push-уведомлений и выберем наиболее приемлемый вариант

A. Элементарная система рассылки на основе сервиса Amazon Simple Notification Service

Для публикации push-уведомлений легче всего использовать Amazon SNS Management Console, для приема - готовое тестовое мобильное приложение, предоставляемое Amazon [15] , позволяющее принимать push-уведомления.

Для организации элементарной системы рассылки необходимо [16]:

1. Создать учетную запись Amazon Web Services

Необходимо указать номер кредитной карты.

2. Создать учетную запись Google

В Google Developers Console необходимо создать новое приложение Google API project и получить project id, project number. Далее необходимо сгенерировать Server API key, необходимый серверу публикаций для связи с GCM. Мобильному приложению для регистрации в GCM необходим project number.

3. Скачать и установить тестовое мобильное приложение, получить registration ID от GCM.

4. В SNS Management Console создать новое приложение для push-платформы GCM с указанием Server API key.

5. Создать Endpoint с указанием registration ID.

Достоинства:

- простая и удобная web-консоль (ничего лишнего).

Недостатки:

- необходимость двух учетных записей;
- ручная регистрация на сервере публикаций - мобильное приложение при запуске не регистрируется на сервере публикаций – необходимо добавить код для регистрации;
- отправленные уведомления не сохраняются;
- отсутствует планировщик.
- native/rich – уведомления;

B. Элементарная система рассылки на основе сервиса Microsoft Azure Notification Hubs

Публикацию push-уведомлений самым простым способом можно осуществлять, запустив скрипт в созданной мобильной службе (Mobile Service) Azure Management Portal, прием push-уведомлений через - готовое тестовое мобильное приложение.

Для организации элементарной системы рассылки необходимо [18]:

1. Создать учетную запись Microsoft.

Необходимо указать номер кредитной карты.

2. Создать учетную запись Google (см. выше)

3. В Azure Management Portal создать Notification Hub, добавить Server API

key, получить Connection String.

4. Создать тестовое приложение по инструкциям [17] с указанием project number и Connection String. Установить приложение. Приложение сначала зарегистрируется на CGM, потом в Notification Hub.

5. Создать мобильную службу (Mobile Service) в Azure Management Portal.

6. Создать задание (job) в планировщике (scheduler) мобильной службы.

7. Создать скрипт (с указанием Connection String и сообщения) в задании для отправки уведомлений.

Достоинства:

- более функциональная консоль (с планировщиком и аналитикой), чем у Amazon.

Недостатки:

- необходимость двух учетных записей;
- отправленные уведомления не сохраняются;
- необходимость скрипта.

C. Элементарная система рассылки на основе сервиса от компании Urban Airship

Публикацию push-уведомлений самым простым способом можно осуществлять из web-консоли Urbain Airship, прием push-уведомлений через - готовое тестовое мобильное приложение.

Для организации элементарной системы рассылки необходимо [18]:

1. Создать учетную запись Urban Airship.

2. Создать учетную запись Google (см. выше)

3. Создать Web – приложение в консоли Urban Airship, добавить Server API key, получить ключ для мобильного приложения.

4. Скачать тестовое приложение, добавить project number и ключ.

5. Запустить мобильное приложение. Приложение сначала зарегистрируется в CGM, потом на сервере Urban Airship.

Достоинства:

- удобная и функциональная web-консоль с

планировщиком и аналитикой;
- отправленные уведомления сохраняются.

Недостатки:
- необходимость двух учетных записей;
- переусложненное тестовое приложение.

D. Элементарная система рассылки на основе сервиса от компании Parse

Публикацию push-уведомлений самым простым способом можно осуществлять из web-консоли Parse, прием push-уведомлений через - готовое тестовое мобильное приложение. Для организации элементарной системы рассылки необходимо [19]:

1. Создать учетную запись Parse.
2. Создать Web – приложение в консоли Parse, получить APP_ID и CLIENT_KEY для мобильного приложения.
4. Скачать тестовое приложение, либо создать новое. Добавить APP_ID и CLIENT_KEY.
5. Запустить мобильное приложение. При первом запуске приложение регистрируется на сервере Parse.

Достоинства:
- удобная и функциональная web-консоль с планировщиком и аналитикой;
- одна учетная запись;
- быстро создаваемое тестовое приложение.

E. Элементарная система рассылки на основе push-бэкенда Pushwoosh

Для отправки push-уведомлений легче всего использовать встроенную консоль. Для оценки функциональных возможностей предлагается готовое тестовое мобильное приложение. Для организации элементарной системы рассылки необходимо [20]:

1. Создать учетную запись Pushwoosh.
2. Создать учетную запись Google (см. выше)
3. Создать Web – приложение в консоли Pushwoosh, добавить Server API key.
4. Скачать тестовое приложение, добавить project number и номер web-приложения.
5. Запустить мобильное приложение.

Достоинства:
- удобная и функциональная web-консоль с планировщиком и аналитикой.

Недостатки:
- необходимость двух учетных записей;
- переусложненное тестовое приложение.

F. Элементарная система рассылки на основе push-бэкенда PushOver

Для организации элементарной системы рассылки необходимо:

1. Получателям уведомлений - скачать платное приложение, зарегистрироваться на сервере PushOver, получить по email уникальный User Key, необходимый для подписки на уведомления.
2. Приложение – отправитель уведомлений, формирователь запросов, зарегистрированное на сервере PushOver для получения уникального Application API token.

Достоинства:
- не нужно разрабатывать мобильное приложение.

Недостатки:
- платное мобильное приложение;
- отсутствие консоли, планировщика, аналитики.

G. Элементарная система рассылки на основе сервера с открытым кодом Uniqush

Для организации элементарной системы рассылки необходимо:

1. Разместить сервер в облаке (напр. Amazon EC2).
2. Создать учетную запись Google (см. выше).
3. Создать мобильное приложение (предоставляются библиотеки и тестовые примеры).
4. Создать приложение – отправитель уведомлений.

Достоинства:
- открытый код сервера;
- одна учетная запись.

Недостатки:
- плата за размещение сервера в облаке;
- отсутствие консоли, планировщика, аналитики.

H. Выбор сервиса push-уведомлений для реализации модели

Наиболее просто и удобно реализовать элементарную систему рассылки на основе сервиса Parse.

Сервис от Amazon, хоть и обладает довольно простой консолью, но приложение необходимо дорабатывать (дописывать код регистрации на сервере публикаций). Сервис от Microsoft сложен, консоль неудобна. Сервисы Pushwoosh и UrbanAirship обладают удобной и функциональной консолью, но тестовые приложения необходимо сильно дорабатывать (убирать лишний код). Uniqush сложен в использовании. Несмотря на открытость кода - придется платить деньги за размещение сервера в облаке. Pushover неудобен платным мобильным приложением и отсутствием консоли.

V. РЕАЛИЗАЦИЯ МОДЕЛИ НА ОСНОВЕ PARSE

.

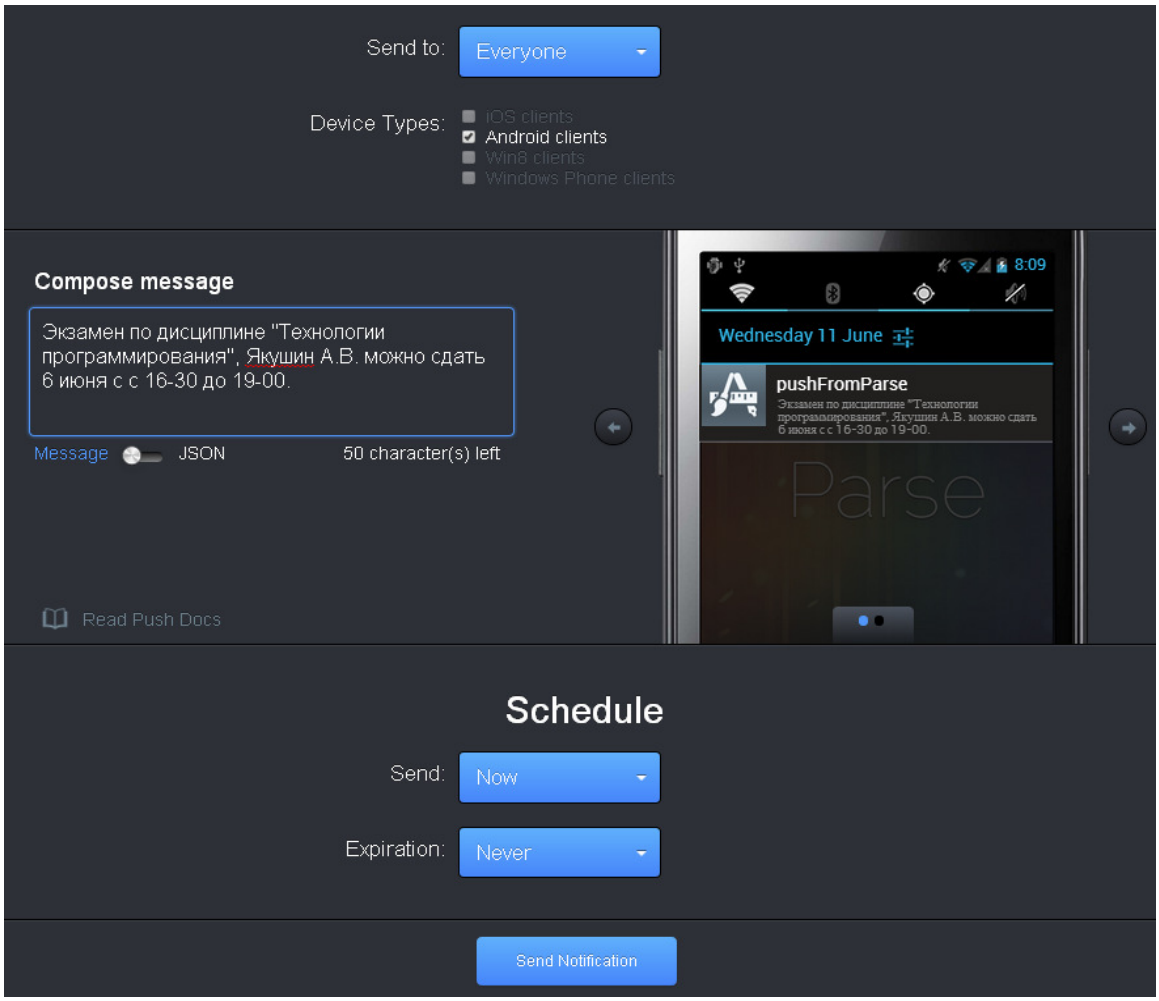

Рисунок 8. Интерфейс публикации уведомлений

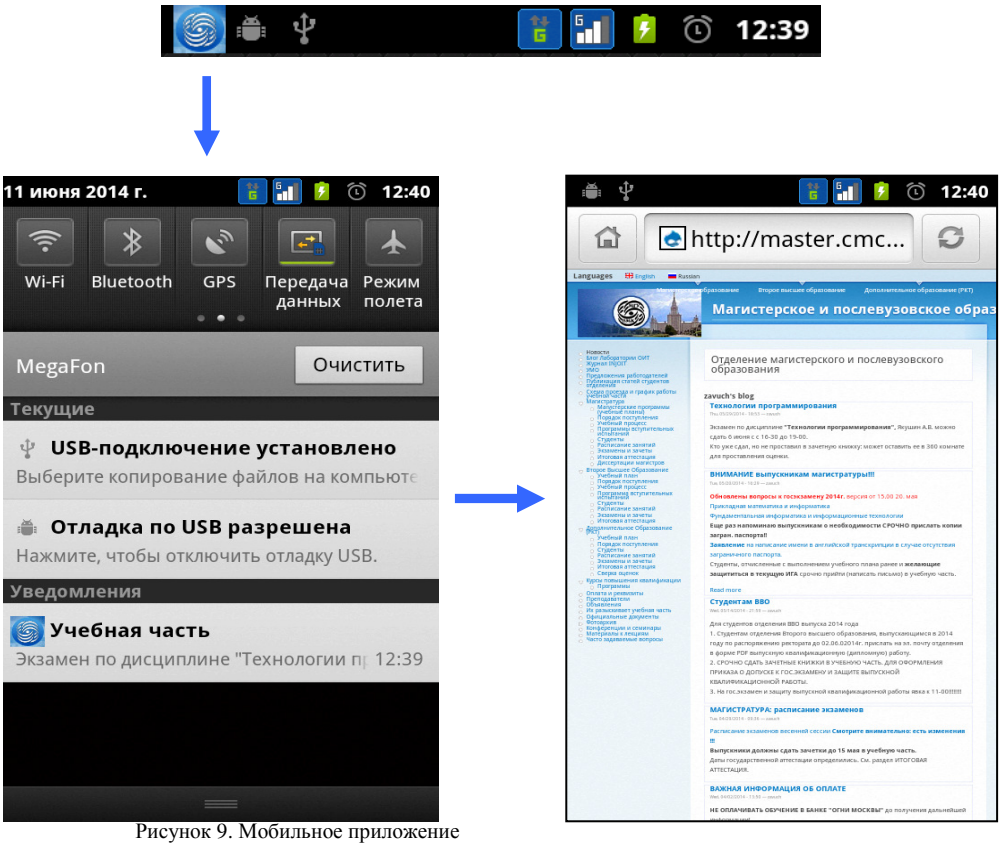

Рисунок 9. Мобильное приложение

Реализация модели регистрации подписчиков и отправки native push-уведомлений на основе сервиса Parse осуществляется крайне просто. Рассмотрим реализацию на простом примере рассылки информации от учебной части отделения магистерского и послевузовского образования факультета ВМК МГУ им. М.В.Ломоносова [21].

Публикация уведомлений осуществляется через консоль Parse (рис. 8). Необходимо создать приложение (application) в консоли Parse, получить APP_ID и CLIENT_KEY для мобильного приложения. В самом простом варианте уведомления будут рассылаться широковещательно всем подписчикам. Отправленные уведомления сохраняются.

Мобильное приложение легко создается по инструкции [19]. При первом запуске приложение регистрируется на сервере Parse. Ключевые строки кода:

```
// Инициализация Parse SDK
Parse.initialize(this, “APP_ID”, “CLIENT_KEY”);
// Определение класса для обработки уведомлений
PushService.setDefaultPushCallback(this,
YourDefaultActivity.class);
```

Parse берет на себя всю работу по взаимодействию с PSP для Android-устройств - GCM (Google Cloud Messaging). Отпадает необходимость в создании аккаунта Google для разработчика мобильного приложения.

Мобильное приложение принимает push-уведомление (рис. 9), при нажатии на уведомление происходит переход на сайт учебной части для более подробного ознакомления с информацией.

I. Заключение

В нашем исследовании был проведен анализ сервисов push-уведомлений. Были рассмотрены сервисы компаний Amazon и Microsoft, сервис от компании Urban Airship, сервис от мобильного бэкенда Parse, сервисы push-бэкендов Pushwoosh и PushOver, а также сервер с открытым кодом Uniqush.

В данной работе были построены информационные модели регистрации подписчиков и посылки native push-уведомлений, посылки rich push-уведомлений, приведена классификация видов рассылок.

В заключительной части работы был произведен анализ сервисов push-уведомлений с целью реализации модели регистрации и посылки native push-уведомлений на их основе. Анализ показал, что наиболее быстро и просто реализовать указанные модели можно на основе сервиса push-уведомлений мобильного фреймворка Parse. На основе выбранного сервиса была выполнена реализация модели регистрации и посылки native push-уведомлений для мобильных устройств под Android.

Библиография

[1] Павлов А.Д., Намиот Д.Е. Системы для поддержки push-уведомлений //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 7. – С. 37-44.

[2] Church, K., & de Oliveira, R. (2013, August). What's up with whatsapp?: comparing mobile instant messaging behaviors with traditional SMS. In Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services (pp. 352-361). ACM.

[3] Намиот Д. Е. Twitter как транспорт в информационных системах //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 1. – С. 42-46.

[4] Chen, C. L., & Raman, T. V. (2008, April). AxsJAX: a talking translation bot using google IM: bringing web-2.0 applications to life. In Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A) (pp. 54-56). ACM.

[5] Namiot, D., & Sneps-Sneppe, M. (2013). Geofence and Network Proximity. In Internet of Things, Smart Spaces, and Next Generation Networking (pp. 117-127). Springer Berlin Heidelberg.

[6] Namiot, D., & Sneps-Sneppe, M. (2013, May). Local messages for smartphones. In Future Internet Communications (CFIC), 2013 Conference on (pp. 1-6). IEEE.

[7] Namiot D. GeoFence services //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 9. – С. 30-33.

[8] Dilger D. E. Inside iOS 7: iBeacons enhance apps' location awareness via Bluetooth LE //ed: Apple Insider.–2013. – 2013.

[9] Намиот Д. Е. Мобильные Bluetooth теги //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 5. – С. 17-23.

[10] Aerogear http://aerogear.org/docs/specs/aerogear-server-push/#overview Retrieved: Jun, 2014

[11] Byron, A., Berry, A., Haug, N., Eaton, J., Walker, J., & Robbins, J. (2008). Using Drupal. " O'Reilly Media, Inc.".

[12] Гурьев Д. Е., Намиот Д. Е., Шнепс М. А. О телекоммуникационных сервисах //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 4. – С. 13-17.

[13] Namiot D. Network Proximity on Practice: Context-aware Applications and Wi-Fi Proximity //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 3. – С. 1-4.

[14] Намиот Д. Е. Персональные Bluetooth теги //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 3. – С. 35-39.

[15] Amazon SNS. URL: http://aws.amazon.com/sns/ Retrieved: Jul, 2014

[16] Getting Started with GCM. URL: http://docs.aws.amazon.com/sns/latest/dg/mobile-push-gcm.html Retrieved: Jul, 2014

[17] Get started with Notification Hubs. URL: http://azure.microsoft.com/ru-ru/documentation/articles/notification-hubs-android-get-started Retrieved: Jul, 2014

[18] Android: Getting Started with Push. URL: http://docs.urbanairship.com/build/android.html Retrieved: Jul, 2014

[19] Android Push Notifications. URL: https://parse.com/tutorials/android-push-notifications Retrieved: Jul, 2014

[20] Native Android SDK. URL: http://www.pushwoosh.com/programming-push-otification/android/native-android-sdk-integration/ Retrieved: Jul, 2014

[21] Магистратура ВМК МГУ http://master.cmc.msu.ru/ Retrieved: Jul, 2014

Information systems based on push-notifications

Pavlov A.D., Namiot D.E.

Abstract—**This paper deals with the problem of building information models based on push-notifications to mobile clients. The aim is to analyze the possibility of construction of information systems based on push-notifications without programming. The ultimate goal - is to build a certain analogue of specialized CMS (Content Management System), focused on the dissemination of information using server notifications. The article presents a basic implementation of the model for registration of subscribers and mailing push-notifications.**

Keywords— **push, notifications, mobile client, CMS.**