# Системы для поддержки push-уведомлений

Павлов А.Д., Намиот Д.Е.

***Аннотация*—Настоящая работа посвящена исследованию возможности построения информационных сервисов на основе push–уведомлений для мобильных клиентов. Конечной целью работы являются модели построения прикладных систем на основе push-уведомлений без программирования. В данной статье, которая является первой частью публикуемого исследования, приводится анализ существующих сервисов push–уведомлений. На основе этого анализа будет предложен набор информационных моделей, использующих push-уведомления. Также мы представим базовую реализацию модели регистрации подписчиков и рассылки push-уведомлений на основе выбранного сервиса.**

***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 последнее время совершенно очевидным является тренд, в котором мобильные абоненты отказываются от традиционных для телекома уведомлений (SMS, MMS) в пользу сообщений от разного рода мессенджеров [1]. Вместе с тем, SMS, например, уже в течение достаточно долгого времени используются как транспорт в различного рода информационных системах. SMS уведомления – это довольно удобный для пользователя и простой для реализации метод рассылки уведомлений. В данном случае мы имеем в виду не рекламные рассылки, а именно уведомления (подписки) из информационных сервисов, системы вопрос-ответ и т.п. Что заменит (может заменить) SMS в такого рода системах?

В работе [2], например, рассматривается использование Twitter, как транспортного уровня для информационных систем. В данной работе, в качестве механизма доставки сообщений рассматриваются push-уведомления. 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 приложение.

Push-уведомления широко используются в мобильных разработках. Например, в работах, выполненных в лаборатории ОИТ [3][4].

Технология Push (server push) описывает один из способов доставки контента мобильным пользователям посредством сети Интернет. В основе данной технологии лежат две модели – клиент-сервер (client/server) и публикатор/подписчик (publisher/subscriber). Мобильный клиент (subscriber) при помощи установленного приложения подписывается на рассылку push-уведомлений, а сервер публикаций (publication server), по наступлению определенного события или по инициативе публикатора (publisher), осуществляет рассылку подписавшимся клиентам. Данная технология применяется для рассылки разного рода информации (сообщения, купоны, новости, прогнозы погоды, реклама, обновление ПО и т. д.).

Важно, что push–уведомления позволяют удаленным серверам уведомлять пользователей о наступлениях событий, даже если приложение неактивно.

Push–уведомления экономят заряд батареи, не требуют постоянного соединения с удаленным сервером; они дешевле SMS, но для их доставки необходимо соединение с интернетом. Экономичность push–уведомлений создается за счет введения промежуточного звена между сервером - отправителем и приложением – получателем. Этим звеном являются службы уведомлений PSP (Push Service Provider):

- GCM (Google Cloud Messaging) для мобильных устройств под Android;
- APNS (Apple Push Notification Service) для мобильных устройств под iOS;
- MPNS (Microsoft Push Notification Service) для мобильных устройств под Windows.

Эти службы поддерживают очень экономичное соединение с мобильными устройствами и осуществляют доставку push–уведомлений. ОС мобильного устройства перенаправляет уведомление соответствующему приложению. В приложениях под Windows Phone уведомления появляются в верхней части экрана. В приложениях под Android и iOS

Статья получена 15 июня 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)

уведомления появляются в верхней части экрана в панели уведомлений.

Но это именно программирование. В данном же исследовании рассматривался вопрос создания информационных систем именно без программирования (или с минимальным объемом такового). В идеале, это должна быть некоторая content management system (CMS) a-la Drupal [5], 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.

Адресатом подобного рода сервиса могут являться, например, учебные заведения, которые смогут организовать современное взаимодействие с учащимися, предприятия торговли (retail) для уведомления покупателей и т.д. Очевидно, что число возможных потребителей больше, чем доступное число разработчиков. Иными словами, такой сервис будет явно востребован и может иметь явную коммерческую направленность.

До недавнего времени рассылка push–уведомлений была возможна только для мобильных клиентов единой платформы. В настоящее время имеется множество сервисов, осуществляющих рассылку миллионам мобильных клиентов на различных платформах. Рассмотрению таких сервисов и посвящена данная статья.

Работа выполнялась в рамках исследований, проводимых в лаборатории ОИТ [6].

## II. Сервис push-уведомлений от компании Amazon

Amazon Simple Notification Service (Amazon SNS) – быстрый и гибкий сервис, осуществляющий организацию публикаций push-уведомлений и их доставки подписавшимся на определенную тему (Topic) подписчикам (Subscribers). Структура сервиса приведена на рисунке 1.

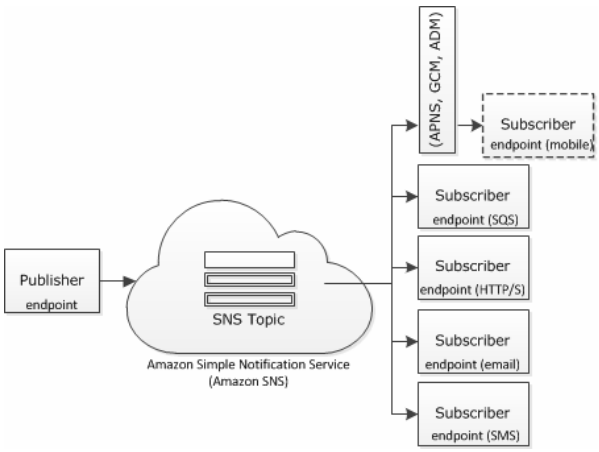


Рисунок 1. Amazon SNS

Основные характеристики [7]:

В качестве адресатов уведомления можно использовать следующие типы клиентов:

- очередь сообщений Amazon Simple Queue Service (Amazon SQS);
- конечная точка HTTP/S;
- электронная почта (email);
- устройство с поддержкой SMS;
- мобильные устройства (mobile) на платформах iOS, Android и Kindle Fire.

Сервис поддерживает индивидуальную и сегментную (групповую) рассылку. Для доставки уведомлений мобильным устройствам на платформах iOS, Android and Kindle Fire Amazon SNS использует средства сервисов push-уведомлений Apple Push Notification Service (APNS), Google Cloud Messaging for Android (GCM) и Amazon Device Messaging (ADM), соответственно.

Конечные пользователи могут публиковать уведомления с помощью следующих средств:

- встроенной консоли Amazon Web Services (AWC) Management Console;
- интерфейса командной строки;
- приложения, использующего Amazon SNS API.

Возможна одновременная рассылка разных уведомлений различным мобильным платформам в JSON объекте. Плата не взимается за 1 млн. уведомлений в месяц и 1 млн. запросов к API. Сверх этого лимита оплата составляет 0.5$ за 1 млн. уведомлений, 0.5$ за 1 млн. запросов к API. Для примера, уведомление длиной 256 Кбайт отправляется за 4 запроса к API.

Планировщик сообщений отсутствует.

В качестве средства аналитики (статистики) используется Amazon CloudWatch.

### A. *Amazon CloutWatch*

Amazon CloudWatch – это сервис, позволяющий анализировать, просматривать, сохранять статистику по каждой теме. Примеры – количество опубликованных сообщений, количество доставленных сообщений и т.д. Amazon CloudWatch позволяет устанавливать порог для отслеживаемых параметров. При превышении порога (например, количество не доставленных сообщений) Amazon CloudWatch посылает уведомление на email.

Для статистики тем Amazon SNS могут использоваться средства:

- консоль Amazon CloudWatch;
- интерфейс командной строки Amazon CloudWatch;
- приложение, использующее Amazon CloudWatch API.

### B. *Организация системы уведомлений*

Для организации минимальной системы push–уведомлений мобильных клиентов на основе Amazon SNS необходимо выполнить следующие действия [7]:

1. Создать клиентское (*AmazonSNSClien*t) платформо – зависимое приложение для одного из поддерживаемых сервисов push–уведомлений (APNS, GCM, ADM) для связи с соответствующим

приложением на мобильной платформе. Формируется уникальный ARN (Amazon Resource Name) идентификатор - *PlatformApplicationArn*.

Необходимая информация:

- атрибут *PlatformPrincipal*, получается от сервиса push – уведомлений: от APNS – сертификат SSL(формат .pem), от GCM – ничего, от ADM - Client ID;

- атрибут *PlatformCredential*, получается также от сервиса push – уведомлений: от APNS - application private key в формате .pem, от GCM - Server API key, от ADM - Client secret.

2. Создать конечную точку (*EndPoint*), характеризующую удаленное мобильное устройство и мобильное приложение.

Необходимая информация:

- *PlatformApplicationArn*

- Token, формируемый сервисом push – уведомлений для мобильного устройства и мобильного приложения: от APNS - device token, от GCM/ADM - registration ID.

Данное действие возвращает атрибут *EndpointArn*. Далее можно уже использовать этот атрибут для публикации (publish) сообщений, либо для подписки (subscribe action) к теме (topic).

3. Создать тему для сообщений. В результате – *TopicArn*.

4. Подписка на тему - subscribe action. Это то, что выполняет клиент. Необходимая информация:

- протокол (protocol)

- http/s – JSON сообщение в теле HTTP/S POST запроса.

- email - SMTP

- email-json – JSON сообщение посредством SMTP

- sms - SMS

- sqs – JSON сообщений для очереди Amazon SQS

- application – JSON сообщения для мобильного приложения и устройства с EndpointArn.

- возможные типы конечной точки (Endpoints) -

- для протокола HTTP/S - URL "http://" или "https://"

- для протоколов email и email-json – адрес email.

- для sms протокола – номер телефона с возможностью SMS.

- для sqs протокола – ARN очереди Amazon SQS.

- для протокола application - EndpointArn мобильного приложения и устройства.

- TopicArn – ARN темы для подписки

5. Публикация сообщений – publish action. Посылка сообщения всем конечным точкам, подписавшимся на тему. Формат сообщения зависит от протокола связи с конечной точкой. Передаваемое сообщение должно быть не более 256 Кбайт, кодировка UTF–8.

## III. Сервис push - уведомлений от компании Microsoft

Windows Azure Notification Hubs – гибкий и масштабируемый облачный сервис, организующий отправку push–уведомлений и доставку их мобильным клиентам различных платформ [8]. Сервис поддерживает индивидуальную и сегментную (групповую) рассылку. Подписчики могут быть сегментированы с помощью тегов. Из мобильных платформ поддерживаются Windows Phone, iOS, Android. Структурная схема представлена на рисунке 2.

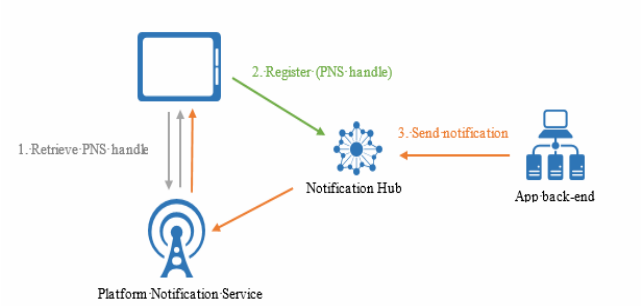


Рисунок 2. Структура Windows Azure Notification Hubs [8]

Из других характеристик:

- наличие механизма шаблонов (Templates) для одновременной рассылки уведомлений разным мобильным платформам;

- публикация уведомлений из портала Microsoft Azure (при помощи скрипта) или со стороны бэкенда (.Net, .Node.js, REST API);

- сбор статистики с выбором метрик из портала Microsoft Azure или через REST API;

Сервис поддерживает планировщик сообщений. Бесплатный пакет включает 500 активных устройств на пространство имен, 100 000 уведомлений/месяц (3 333 уведомлений/сутки), 500 000 вызовов API в месяц.

### A. *Функционирование сервиса*

Мобильное устройство получает *PNS handle* (например, WNS ChannelURI, APNS device token, GCM registrationId).

2. Далее мобильное устройство регистрируется в *Notification Hub* с указанием *PNS handle*. Для разграничения мобильных устройств (сегментация пользователей) используются теги или наборы тегов. Также возможна посылка шаблона уведомления (JSON, XML). Шаблон определяет формат принимаемого пользователем уведомления.

3. Серверное приложение осуществляет рассылку. При рассылке возможно указание тегов или наборов тегов для ограничения рассылки.

На рисунке 3 показана схема рассылки групповых уведомлений.

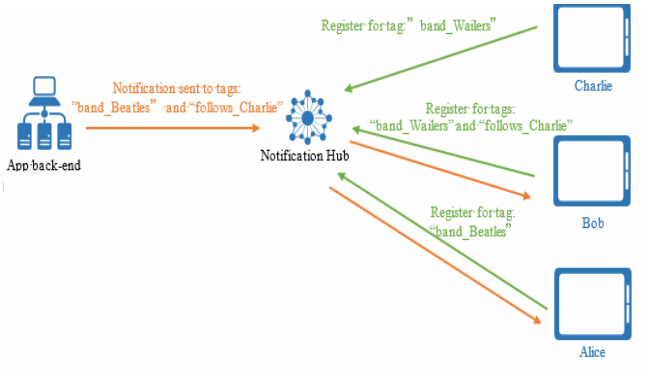


Рисунок 3. Сегментные уведомления в Windows Azure Notification Hubs [8]

## IV. Сервис push-уведомлений от компании Urban Airship

Компания Urban Airship (Portland, USA) специализируется на предоставлении сервиса push–уведомлений с учетом пользовательских предпочтений и персональной информации (мобильные платформы, географическое положение и т.д.) с использованием стратегий вовлеченности и мобильной аналитики [9].

Сервис поддерживает индивидуальную и сегментную (групповую) рассылку. Подписчики могут быть сегментированы с помощью тегов. Из мобильных платформ поддерживаются Android, iOS, Windows 8, Blackberry.

Из других характеристик сервиса:

- native/rich push–уведомления;
- web - консоль с планировщиком и аналитикой;
- web - компоновщик rich push-уведомлений с возможностью предпросмотра;
- сохранение отправленных уведомлений;
- наличие планировщика;
- REST API для отправки уведомлений.

Бесплатный аккаунт обеспечивает рассылку до 1 млн./мес. native push–уведомлений (свыше - $0.001 каждое), каждое rich push–уведомление стоит $0.0025.

Термин rich относится к содержимому сообщения. Отправить пользователю можно произвольный контент, указав соответствующий mime/type.

## V. Сервис push-уведомлений от компании Parse

Parse (Facebook, Inc.) - облачный бэкенд для мобильных приложений (mBaaS: mobile-backend-as-a-service). Предлагает для разработчиков мобильных приложений услуги удалённого сервиса для хранения и обработки данных, а также сервис push-уведомлений [10]. Это на самом деле платформа для мобильной разработки, где уведомления есть только малая часть пакета.

Сервис поддерживает индивидуальную и сегментную (групповую) рассылку. Подписчики могут быть сегментированы с помощью каналов. Из мобильных платформ поддерживаются Android, iOS, Windows Phone.

Из других характеристик сервиса:

- рассылка уведомлений через web console, REST API;
- push composer – предпросмотр уведомлений;
- наличие планировщика.

Бесплатный пакет включает неограниченное число уведомлений для 1 млн. получателей. Оплата свыше - 0.05$ за 1000 получателей.

## VI. Push–бэкенд Pushwoosh

Pushwoosh – это push–бэкенд от компании Arello Mobile [11]. Сервис поддерживает индивидуальную и сегментную (групповую) рассылку. Подписчики могут быть сегментированы с помощью тегов. Из мобильных платформ поддерживаются Android, iOS, Windows Phone, Blackberry.

Из других характеристик сервиса:

- Geozones – уведомления подписчиков при вхождении в определенную зону;
- rich push–уведомления;
- multi language - уведомления;
- мощная статистика;
- планировщик;
- REST API.

Бесплатный пакет включает в себя 1000000 устройств, 5 приложений, rich push–уведомления.

## VII. Push–бэкенд с готовым мобильным приложением PushOver

PushOver - платформа для приема/отправки push–уведомлений [12]. Серверная часть представлена REST API для прямого или группового уведомления клиентов. Мобильная часть представлена приложением (Android / iOS) для приема, чтения и хранения полученных уведомлений. Возможна интеграция PushOver с приложением, сайтом, серверным процессом и т.д.

Для получения уведомлений необходимо скачать клиентское приложение, зарегистрироваться на сервере PushOver, получить по email уникальный User Key, необходимый для подписки на уведомления.

Каждое приложение, осуществляющее посылку сообщений, должно быть зарегистрировано на сервере PushOver для получения уникального Application API token. Зарегистрированное приложение имеет возможность посылки 7500 сообщений в месяц бесплатно. Все, что выше оплачивается из расчета 50$ за блок из 10000 сообщений.

Сообщение отправляется HTTPS POST запросом на адрес *https://api.pushover.net/1/messages.json* со следующими параметрами:

- token (обязательно) - Application API token приложения

- user (обязательно) – User Key или Group key пользователя или группы пользователей
- message (обязательно) – сообщение

Опциональные параметры:
- device – имя устройства пользователя для прямого сообщения одному устройству одного пользователя (у одного пользователя может быть несколько устройств)
- title – заголовок сообщения
- url – дополнительный URL, показываемый вместе с сообщением.
- url_title – заголовок дополнительного URL
- priority – приоритет сообщения – определяет то, как сообщение будет представлено пользователю:

0 – нормальный приоритет – звук, вибрация, установленные пользователем.

-1 – низкий приоритет – без звука, без вибрации.

1 – высокий приоритет – звук и вибрация всегда (при соответствующих настройках устройства)

2 – высокий приоритет – для случаев, когда необходимо, чтобы пользователь подтвердил, что он прочитал сообщение. Сообщение посылается с определенной периодичностью (параметр retry) определенный отрезок времени (параметр expire), опционально можно прислать URL для подтверждения просмотра сообщения (параметр callback).

- timestamp – установленная отправителем отметка времени. Иначе используется время приема сообщения сервером PushOver.
- sound – имя звукового сигнала, поддерживаемого устройством – перекрывает установленный сигнал.

Параметры помещаются в тело запроса в виде пар ключ-значение.

Длина сообщения не должна превышать 512 символов, включая заголовок. Дополнительные URL не более 512 символов, заголовки – не более 100. Клиентское приложение автоматически находит и выделяет гиперссылки в тексте сообщения. Ответ присылается в формате JSON. При успешном запросе – HTTP 200 с JSON объектом (status = 1). В случае ошибок – HTTP 4xx с JSON объектом (status не равен 1).

### A. *Доставка и прием сообщений*

Организация процесса доставки и хранения уведомлений выглядит следующим образом.

1. Приложение (в т.ч. сервер, плагин), используя API, посылает уведомление серверу PushOver поверх HTTPS с использованием API token, User Key или Group Key.

2. Сервер PushOver ставит в очередь уведомление, сохраняя в базе данных.

3. Сервер PushOver соединяется с GCM или APNS с использованием SSL и передает очередное уведомление. При успешной доставке - сообщение в БД помечается как доставленное.

4. GCM или APNS доставляют сообщение конечному устройству посредством собственного, постоянного SSL соединения.

- На мобильных устройствах под Android уведомления принимаются приложением Pushover, сохраняются в локальной БД и выдает индикатор приема.
- На мобильных устройствах под iOS - если приложение Pushover запущено, то оно сохраняет сообщение в локальной БД и выполняет уведомление. Если приложение не запущено, то iOS сохраняет уведомление и выдает сигнал тревоги. Сообщение не будет принято приложением Pushover, пока приложение не будет запущено и синхронизировано с сервером.

5. При запуске мобильного приложения Pushover происходит соединение с сервером Pushover и прием не принятых сообщений.

6. Мобильное приложение сообщает серверу Pushover номер принятого сообщения из БД с самым большим ID. Сервер удаляет все сообщения с меньшими ID. Принятые сообщения остаются только в мобильном устройстве.

Доставленные сообщения на сервере не хранятся. Не доставленные сообщения хранятся на сервере в течение 21 дня.

## VIII. Сервер push–уведомлений с открытым кодом Uniqush

Uniqush - бесплатный сервер с открытым кодом для организации рассылки push - уведомлений мобильным клиентам на платформах iOS, Android and Kindle Fire [13]. Сервер работает с базой данных Redis. Для доставки уведомлений сервер использует средства APNS, GCM, ADM.

Для осуществления рассылки уведомлений необходимо послать HTTP запрос серверу (процессу) Uniqush. Далее сервер выполнит другие HTTP запросы соответствующим провайдерам push–уведомлений (Push Service Provider) в зависимости от мобильной платформы. Push Service Provider осуществит доставку уведомлений мобильным клиентам.

Основные операции:
- соединение с сервером. Сервер Uniqush - отдельное, самостоятельное приложение, прослушивающее определенный порт. Для отправки серверу сообщения необходим HTTP POST запрос. Типы запросов:

- добавление/удаление Push Service Provider. Серверу необходимо сообщить о типах провайдеров (GCM, APNS.) и информацию для связи с ними - authentication token для GCM; private/public keys для APNS.
- подписка/удаление подписки на сервис. Для подписки на сервис серверу необходима информация: имя подписчика, имя сервиса, данные о мобильном устройстве (тип платформы, идентификационная информация). Данная информация посылается в виде HTTP POST запроса серверу напрямую или через

некоторый прокси.

- рассылка уведомлений (запрос push). Серверу необходимо сообщить имя сервиса, имя подписчика, данные для рассылки.

Некоторые параметры запросов:
- service - имя сервиса;
- subscriber - имена подписчиков (от одного и больше);
- msg сообщение (опционально);
- ttl -время жизни в секундах - время хранения сообщения на стороне провайдера push - уведомлений при неактивном устройстве (опционально);
- badge - значок (опционально);
- img - картинка (опционально);
- sound - звук (опционально).

Как минимум один из опциональных параметров должен быть указан, иначе сообщение будет пустым и не будет отправлено.

Для мобильных клиентов под Android и Kindle Fire сообщение будет получено в виде Intent пар ключ - значение. Сообщение (msg), значок (badge), Картинка (img), звук (sound) и параметры пользователя будут положены в Intent в виде пар ключ - значение. Для извлечения пар используется метод *getExtras()* Intent. Для клиентов под iOS сообщения посылаются в виде JSON сообщений.

Конфигурация сервера осуществляется через конфигурационный файл *uniqush-push.conf*.

## IX. Анализ рассмотренных сервисов push–уведомлений

Произведенный обзор сервисов push-уведомлений позволяет сделать следующие выводы и обобщения:

1. Push-уведомления бывают двух видов:

- native push-уведомления - предоставляются на уровне ОС, осуществляют доставку только текстовой информации;
- rich push-уведомления - расширение первого вида уведомлений – осуществляют доставку HTML страничек, картинок, видео, опросных страничек (survey push).

2. Различают следующие виды рассылки push-уведомлений:

- индивидуальная рассылка (Unicast) – уникальные сообщения отдельным подписчикам;
- широковещательная рассылка (Broadcast) – одинаковые сообщения всем подписчикам одного приложения;
- сегментная (таргетированная) рассылка (Multicast) – на основе информации, полученной от клиентов во время регистрации, во время использования мобильного приложения и проведенной аналитики данной информации с целью оптимизации рассылки.

3. Сегментация может осуществляться с учетом:

- пользовательских тегов;
- попадания значения параметра клиента в определенный интервал (прим. - временной период, возрастной период), логические выражения с тегами;
- предпочтений, интересов клиентов;
- географического положения, временной зоны;
- поведения клиентов (например, уведомление клиентов давно не открывавших приложение);
- версии мобильного приложения, ОС, типа устройства.

4. Основная цель успешной коммерческой рассылки push–уведомлений – доставить уведомление нужному, заинтересованному подписчику в нужное время и место.

5. Собранная публикатором от подписчиков информация позволяет проводить анализ для оценки того, как подписчики используют приложение. Аналитика (статистика) позволяет определить:

- количество клиентов, получивших уведомления
- количество клиентов, открывших уведомление
- количество клиентов, удаливших приложение и т.д.

6. Публикация уведомлений осуществляется в виде HTTP/S POST запроса. Запрос может содержать специальные заголовки, в теле – текст в формате JSON / XML. Текст сообщения включает данные (*payload)*, специфичные для каждой платформы. *Payload* информирует мобильное приложение о том, каким образом информировать пользователя о наступлении события (звук, вибрация и т.д.)

7. Результаты анализа сервисов push-уведомлений представлены на рисунке 4.

## X. Заключение

В данной статье был проведен анализ сервисов push-уведомлений. Были рассмотрены сервисы компаний-гигантов 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. Для сравнения сервисов использовалась выработанная общая методика. В следующей части исследования будет предложен набор информационных моделей, использующих push-уведомления.

| Характерис-тики / Сервис | Мобильные платформы | Виды рассылки | Средства публикации | Бесплатный пакет (в месяц) | Доп. возможности и особенности |
|---|---|---|---|---|---|
| **Amazon SNS Mobile push** | Android, iOS, Kindle Fire | -**Unicast** -**Broadcast** -**Multicast** | Web-консоль, REST API, SDK | 1 млн. увед. + 1 млн. запр. к API | Сбор статистики, **native** push-уведомления |
| **Windows Azure Notification Hubs** | Android, iOS, Windows Phone | | | 100 000 увед. + 500 000 запр. к API | Сбор статистики, планировщик, **native** push-уведомления |
| **Urban Airship** | Android, iOS, Windows Phone, Blackberry | | | 1 млн. native push-увед. | Сбор статистики, планировщик, **native/rich** push-уведомления, **geo location, iBeacons** |
| **Parse** | Android, iOS, Windows Phone | | | Неог. число увед., 1 млн. получателей | Сбор статистики, планировщик, **native** push-уведомления |
| **Pushwoosh** | Android, iOS, Windows Phone, Blackberry | | | Неог. число увед., 1 млн. получателей, 5 приложений, rich push-увед. | Сбор статистики, планировщик, **native/rich** push-уведомления, **geozones** |
| **PushOver** | Android, iOS | | REST API | 7500 увед. на приложение | Готовое платное мобильное приложение, **native** push-уведомления |
| **Uniqush** | | | | Без ограничений | Открытый код, **native** push-уведомления |

Рисунок 4. Сервисы push-уведомлений

## Библиография


[1] 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.

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

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

[4] Sneps-Sneppe, M., & Namiot, D. (2013). Spotique: A New Approach to Local Messaging. In Wired/Wireless Internet Communication (pp. 192-203). Springer Berlin Heidelberg.

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

[6] 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.

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

[8] Notification Hubs. URL: http://azure.microsoft.com/en-us/services/notification-hubs/ Retrieved: Jun, 2014

[9] Urban Airship. URL: http://urbanairship.com/ Retrieved: Jun, 2014

[10] Parse. URL: https://parse.com/ Retrieved: Jun, 2014

[11] Pushwoosh. URL: http://www.pushwoosh.com/ Retrieved: Jun, 2014

[12] PushOver. URL: https://pushover.net/ Retrieved: Jun, 2014

[13] Uniqush: URL: http://uniqush.org/ Retrieved: Jun, 2014

# Backend systems for push notifications

Pavlov A.D., Namiot D.E..

*Abstract*—**This paper considers the possibility of building of information services based on push-notifications to mobile clients. The ultimate aim is to build a model of application systems based on push-notifications without programming. In this article, which is the first part of the published study, we analyze the existing services for push-notifications. On the basis of this analysis we will present a set of information models using push-notifications. Also, we will present a basic implementation of our model with registration of subscribers and mailing push-notifications based on the selected service.**

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