Московский государственный университет им. М.В. Ломоносова

Факультет Вычислительной Математики и Кибернетики

ДИПЛОМНАЯ РАБОТА

АНАЛИЗ И РАЗРАБОТКА СИСТЕМЫ PUSH-УВЕДОМЛЕНИЙ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ GOOGLE Inc.

Работу выполнил студент ВВО:

Павлов Владимир Владимирович

Научный руководитель:
с.н.с., к.ф.-м.н.
Намиот Дмитрий Евгеньевич
Лаборатория ОИТ факультета
ВМК МГУ им. М.В. Ломоносова

Москва, 2013г.

Содержание

Введение

В настоящее время уровень операционных систем мобильных устройств позволяет технологически использовать различные формы получения информации и контента через приложения и во многом уже не зависеть от провайдера телекоммуникационных услуг. Более того, использование открытых информационных технологий в программировании увеличивает конкуренцию на рынке программного обеспечения, что создает предпосылки к повышению прикладного уровня разработок и позволяет пользователю расширить функциональные возможности своих устройств.

Такие возможности для разработки новых направлений сегодня представлены большинством производителей операционных систем для мобильных устройств. При этом основные игроки на рынке (Google, Apple, BlackBerry) пытаются максимально безопасно открыть доступ к телефону извне для связи с абонентом.

Одним из перспективных направлений в этой области можно выделить создание и развитие системы push-уведомлений для непосредственной отправки сообщений, графического материала и прочей информации посредством сети Интернет на приложения мобильного устройства. В этом случае пользователь системы получает возможность целенаправленного одностороннего общения с заинтересованными лицами.

ГЛАВА I. Обзор и анализ существующих технологий push-уведомлений для мобильных устройств

1.1. Описание push-технологии мобильных устройств

Push-технология представляет собой способ уведомления клиентов, который предполагает наличие мобильного устройства с установленным приложением для осуществления подписки к службам на сервере приложений с целью получения информации непосредственно на приложение по совершению определенного события. В данном случае событие является триггером для сервера приложений, который срабатывает на определенные условия, например, в соответствии с установленным расписанием или завершением определенной работы, и осуществляет рассылку требуемой информации клиенту. Реализация данной технологии подразумевает использование клиент-серверной архитектуры.

В настоящее время эта технология является одним из вариантов предоставления контента пользователям мобильных устройств, например, стоимости акций на биржах, прогноза погоды, либо любого другого информационного сообщения, и не требует от пользователя принятия действий в отношении отправителя. Данная технология активно начинает использоваться в силу минимальных необходимых затрат для ее запуска.

Идея развития push-технологии в мобильных устройствах предполагает, что основное приложение находится в неактивном режиме, и призвана прежде всего преодолеть ряд ограничений, с которыми сталкиваются пользователи и разработчики:

1) ограниченный емкостной ресурс системы питания;

2) относительно высокий уровень оплаты услуг за контент провайдерам (SMS, MMS);

3) высокий уровень конкуренции на рынке мобильных операционных систем и как следствие, развитие дополнительных бесплатных сервисов;

4) сложность реализации взаимодействия с телекоммуникационной инфраструктурой.

Данные факторы создают основу для популярности мобильных приложений, которые способствуют развитию инфраструктуры для передачи push - сообщений.

2.2. Push-технология Google Inc

Система облачных сообщений Google (Google Cloud Messaging, далее GCM) - служба, построенная с использованием ресурсов Интернета, которая позволяет организовать рассылку данных со стороннего сервера пользователям устройств с операционной системой Android. Сообщения представляют собой легковесные данные объемом 4 kB, отправляемые сервером при наступлении определенного типа событий. Данная служба управляет возникающими очередями сообщений и доставкой их до целевого приложения.

Общая схема архитектуры push-уведомлений Google для мобильных устройств представлена на рис. 1.

Можно выделить следующие основные особенности GCM-службы [1]:

1) возможность использовать сторонний сервер приложений для отправки данных;

2) приложение на мобильном устройстве в момент получения сообщения может быть неактивным. В случае соответствующей настройки приложения система запустит данное приложение для получения сообщения в фоновом режиме;

3) установленная операционная система Android версии 2.2 и выше;

4) наличие Google-аккаунта для устройств с операционной системой версии ниже Android 4.0.4.

Таким образом, для использования мобильного приложения на стороне клиента не предъявляются высокие требования, что, безусловно, позволяет популяризировать технологию.

Для организации полноценной работы система должна включать в себя следующий минимальный набор компонент:

1) мобильное устройство;

2) сторонний сервер приложений для отправки данных через службу GCM;

3) службу GCM – служба доставки сообщений до приложения клиента.

При работе в системе используются следующие параметры доступа [1]:

1) идентификатор отправителя (Sender ID) – номер проекта, регистрируемый разработчиком в сервисах Google. Используется в момент регистрации Android-приложения для идентификации устройства при отправке сообщения;

2) идентификатор приложения (Application ID) – наименование приложения, которому адресовано сообщение;

3) регистрационный идентификатор (Registration ID) – номер, формируемый GCM сервером для Android-приложения, которое позволяет принимать сообщения. Данный идентификатор используется сторонним сервером приложений для рассылки данных;

4) аккаунт пользователя в системе Google (Google User Account);

5) маркер авторизованного отправителя (Sender Auth Token) – ключ, используемый сервером приложений для доступа к службам Google.

Основные этапы процесса работы данной системы push-уведомления можно выделить в две основные группы: регистрация, отправка и получение сообщения.

Рассмотрим эти группы этапов более подробно [1].

1) Регистрация в GCM.

1.1) В момент регистрации активируется регистрационный Интент (com.google.android.c2dm.intent.REGISTER), который содержит идентификаторы отправителя и приложения (1).

1.2) В случае успешной регистрации GCM сервер сообщает приложению регистрационный идентификатор (2).

1.3) Для завершения регистрации Android-приложение отправляет регистрационный идентификатор серверу приложений (3).

2) Отправка и получение сообщения.

2.1) Сервер приложения отправляет сообщение GCM-серверу (4).

2.2) Google ставит в очередь и сохраняет сообщение, если устройство выключено или недоступно.

2.3) Когда устройство доступно, GCM служба перенаправляет сообщение мобильному приложению (5).

2.4) Специальная служба операционной системы Android (notification service) ретранслирует сообщение целевому приложению.

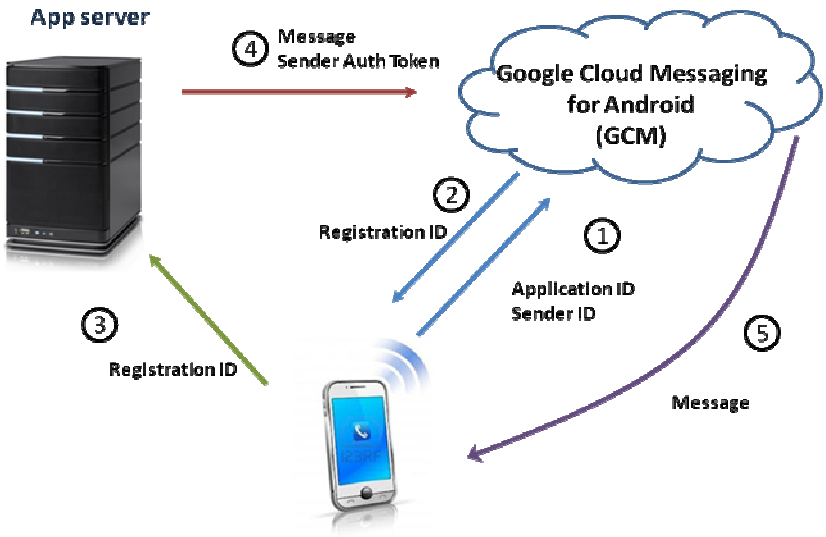

Рис.1. Общая схема архитектуры push-уведомления Google для мобильных устройств

Момент доставки уведомления на мобильное устройство может не совпадать с моментом отправки сообщения GCM службе и получения сервером приложения присвоенного идентификационного номера уведомления, что связано с использование системы Throttling.

Данная система предназначена для предотвращения лавинообразного потока сообщений пользователю и общей оптимизации работы сети и батареи мобильного устройства. Основана она на использовании ограниченного числа токенов для приложения, которые расходуются по мере поступления уведомления. На каждое сообщение система вычитает один токен. Токены в свою очередь пополняются с определенным временным лагом. В случае исчерпания лимита сообщения помещаются в буферную очередь GCM службы, что может создавать временные задержки по доставке уведомлений.

В целом можно отметить, что использование данной архитектуры позволит создать достаточно простую и легко масштабируемую систему уведомлений.

Вся базовая логика взаимодействия с GCM службой предоставляется в специальной библиотеке jar.

1.3. Push-технология компании Apple

Push-технология Apple (далее APN) – служба, разработанная компанией Apple, и использующая постоянно открытое зашифрованное IP соединение для пересылки уведомлений со стороннего сервера Apple-устройствам. Уведомления могут содержать значки, звуки или текст. Сервис был запущен в июне 2009г. При этом каждое уведомление ограничено размером в 256B. Для проверки подлинности push-запросов из iOS-приложения Apple использует цифровые сертификаты с открытым ключом. Если уведомление поступает на устройство с остановленным приложением, появляется сигнал о поступлении новых данных.

Общая схема архитектуры приведена на рис.2.

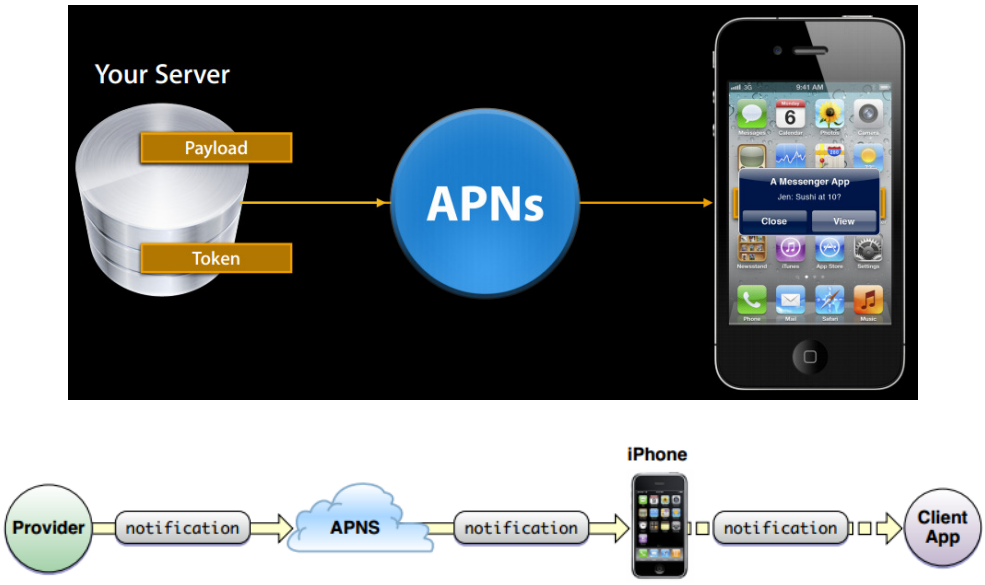

Рис.2. Общая схема архитектуры push-уведомления Apple для мобильных устройств[1]

[1] [2]

В целом схема работы push-технологии Apple совпадает с ранее рассмотренной службой Google и включает в себя следующие этапы [2]:

1) В момент активации приложения через диалоговое окно в iOS-приложении система запрашивает разрешение пользователя на получение уведомлений.

2) Если разрешение получено, iOS-приложение подключается к службе Apple Push Notification (APNs) за строкой уникального идентификатора для этого установленного на устройстве приложения (его можно представить как аналог телефонного номера получателя в традиционном сценарии обмена сообщениями).

3) iOS-приложение передает идентификатор на сервер-приложений.

4) В случае необходимости серверному приложению осуществить отправку push-сообщения, APN проверяет подлинность push-сервера и использует идентификатор для указания получателя сообщения.

5) Если устройство получателя находится в режиме онлайн, оно принимает и обрабатывает сообщение. Если устройство недоступно, сообщение ставится в очередь и доставляется, как только устройство выйдет на связь.

При этом система первоначальной аутентификации устройств и сторонних серверов устроена сложнее и происходит с использованием протокола TLS. В данном случае APN служба после установления соединения с ней направляет на мобильное устройство сертификат, который проходит проверку, после чего сертификат самого устройства перенаправляется для проверки в APN. В случае успешной верификации устанавливается TSL соединение (см. рис. 3) [2].

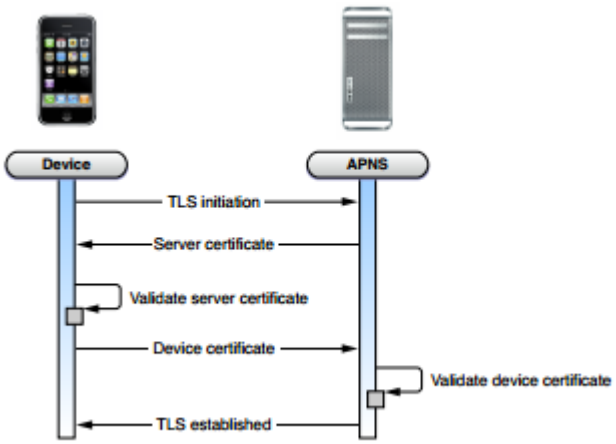

Рис.3. Процесс установления соединения[2]

Каждое уведомление, которое отправляется в службу APN, должно сопровождаться идентификатором устройства, полученного с клиентского приложения. Данный идентификатор расшифровывается для оценки валидности уведомления и получения ID устройства назначения (см. рис 4.) [2].

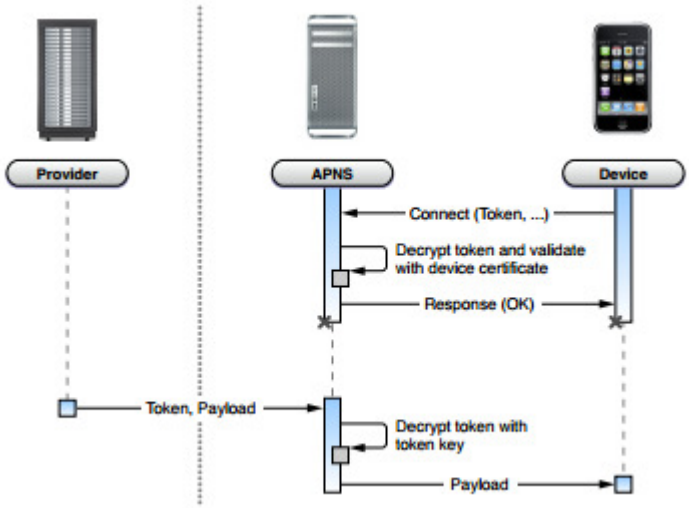

Рис 4. Процесс отправки сообщения[3]

[2] [2]
[3] [2]

1.4. Push-технология BlackBerry

Архитектура системы push-технологии BlackBerry включает в себя инициатора push-уведомлений и устройство BlackBerry с установленным приложением, которое способно принимать такие сообщения. В данной технологии предполагается, что инициатор не дожидается запроса на отправку контента от получателя. Максимальный размер данных не должен превышать 8KB. Компания BlackBerry утверждает, что сообщения мгновенно поступают на приложения пользователя [3].

На диаграмме видно, что архитектура представляет собой клиент-серверное решение (см. рис. 5). Библиотеки серверной и клиентской частей взаимодействуют с различными компонентами системы для обеспечения доставки контента.

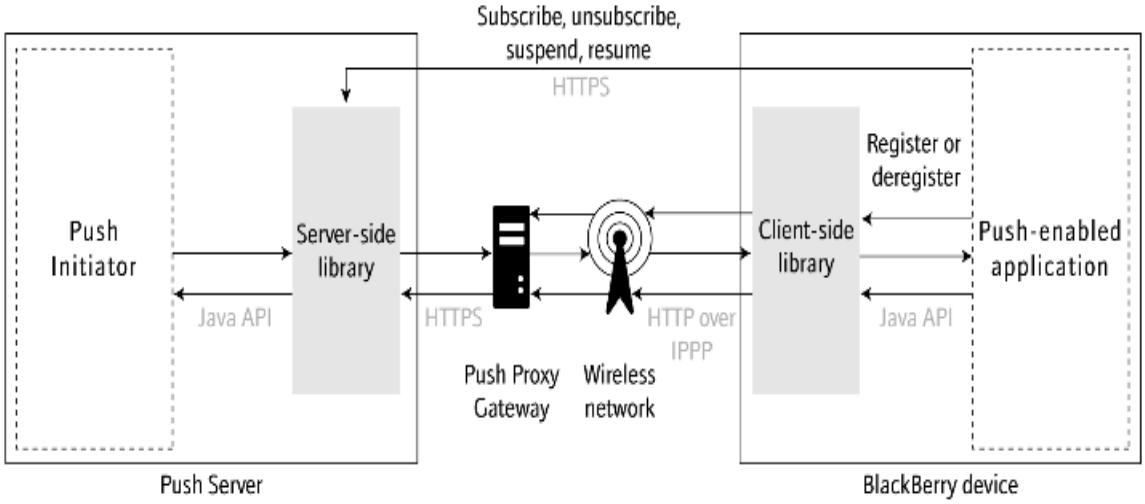

Рис.5. Общая схема архитектуры push-уведомления BlackBerry для мобильных устройств[4]

[4] [3]

Основные компоненты push-технологии BlackBerry[5]

Таблица 1

Компоненты	**Описание**
Push-инициатор (Push Initiator)	Приложение на стороне серверной части, которое создает сообщения-запросы и сообщения-ответы с использованием библиотеки серверной части и передает на PPG.
Библиотека серверной части (Server-side library)	Необходимое программное API для взаимодействия с PPG в формате XML со стороны сервера.
Push прокси-шлюз (Push Proxy Gateway - PPG)	Процессы PPG проталкивают сообщения, которые поступают от инициатора, а также формируют код ответа о доставке. В качестве PPG выступает инфраструктура компании BlackBerry.
Библиотека клиентской части (Client-side library)	Необходимое программное API для взаимодействия с PPG (The BlackBerry® Java® Development Environment 5.0) со стороны клиента.
Клиентское push-приложение (Push-enabled application)	Приложение на клиентском устройстве BlackBerry, которое осуществляет подписку на сервере PPG и прослушивание поступающего контента в фоновом режиме.

[5] [3]

Алгоритм работы системы в целом аналогичен работе рассмотренных ранее push-технологий (рис.6.) и состоит из следующих этапов [3]:

1. поставщик контента (инициатор) отправляет push-запрос;
2. инфраструктура BlackBerry (PPG) возвращает ответ;
3. PPG проталкивает данные в мобильное устройство BlackBerry;
4. мобильное устройство возвращает ответ PPG;
5. ответ-подтверждение направляется инициатору сообщения;
6. подтверждение о полученных данных прочитанного уведомления возвращается на сервер PPG.

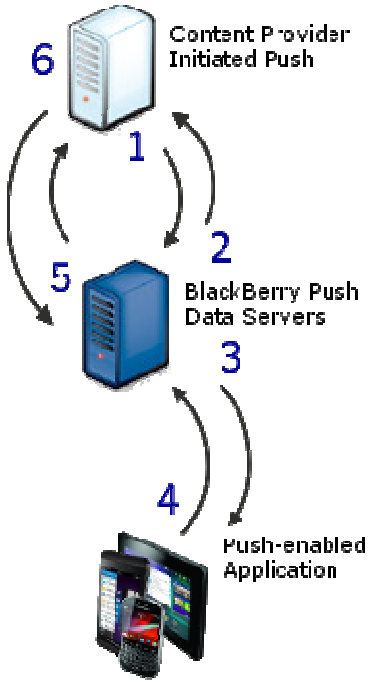

Рис.6. Алгоритм работы службы уведомления BlackBerry[6]

[6] [3]

1.5. Анализ интеграционных возможностей push-технологий

Все рассмотренные технологии предоставляют возможность разработчикам использовать библиотеки, написанные на языке Java, для построения различных архитектур стороннего сервера приложений для одновременной работы с инфраструктурой Google, Apple и BlackBerry.

Взаимодействие со службой APN может быть организовано с помощью зарекомендовавшей себя библиотекой JavaPNS с открытым исходным кодом. Основные методы интерфейса данной библиотеки представляют собой статические методы класса Push для сообщений каждого типа и выполняют преобразование сообщения в формат JavaScript Object Notation (JSON), принимаемый серверами APNs [2].

Компания BlackBerry разработала собственный комплект средств разработки для реализации взаимодействия с собственной инфраструктурой - Push Service SDK, которая предоставляет Java APIs, используемое Push-инициатором для взаимодействия с PPG [3].

Таким образом, возможна реализация легко масштабируемой архитектуры, способной взаимодействовать с платформами основных игроков на рынке мобильных устройств.

1.6 Обзор аналогов действующих push-систем

Идеи развития систем push-уведомлений с расширенными функциональными возможностями для пользователей существуют давно. Первые и наиболее интересные реализации присутствуют на рынке уже не первый год, но в качестве языков программирования в большинстве коммерческих проектов используются HTML, CSS, JavaScript.

Наиболее ярким примером в этой области является программный продукт AppCloud компании Brightcove – для создания мобильных приложений с использованием технологии HTML5. Данная система позволяет дополнительно создавать, устанавливать расписание и определять целевую аудиторию по географическому признаку для рассылки уведомлений служб GCM и APN.

Система push-уведомлений полностью интегрирована с собственной системой аналитики и позволяет получить маркетинговые оценки проводимых кампаний и определить поведение клиентов по активности самого приложения. Более подробная информация представлена на сайте разработчика [6]. Тем не менее, компания объявила о закрытии своего проекта в 2013г.

Другим коммерческим проектом, активно использующим технологии push-систем Google и Apple, является Push Messaging компании Urban Airship [8]. В данном программном продукте реализована та же функциональность, что и в AppCloud за исключением того, что создание мобильного приложения и интеграцию его с системой рассылки пользователю необходимо решать самостоятельно.

В целом на рынке сейчас активно развиваются и другие проекты (например, PushIO, AirBop), которые позволяют через веб-интерфейс либо через веб-службы получить доступ к API серверной части системы push-уведомлений и начать организованную рассылку контента на мобильные приложения.

В качестве основного недостатка большинства реализованных коммерческих систем можно выделить необходимость интегрировать клиентское приложение в общую систему, что создает дополнительные препятствия для заказчика.

Тем не менее, реализация расширенной функциональной части по управлению системой рассылки выгодно выделяет такие проекты для маркетинговых услуг, так как позволяет:

1) ..... сегментировать пользователей по месторасположению и по предпочтениям;
2) ..... осуществлять рассылку в определенные периоды времени;
3) ..... минимизировать затраты на рекламу на каждого клиента;
4) ..... собирать аналитику для изучения поведения клиентов.

Что касается open source решений, то готовых систем, написанных на языке Java, на рынке не представлено. Тем не менее, среди прочих аналогичных проектов можно выделить Uniqush (C++), NuGet(C#), Easy APNs (php), PyAPNS(python), PushSharp(C#). Данные системы позволяют отправлять сообщения с учетом функциональности, которая заложена в соответствующие службы APN, GCM и PPG.

ГЛАВА II. Разработка системы push-уведомлений

2.1. Описание предлагаемой архитектуры системы push-уведомлений

В качестве объекта для разработки архитектуры системы, реализующей функции push – уведомлений, была выбрана за основу базовая технология компании Google Inc. Основными критериями, повлиявшими на решение, в большей степени явились факторы ориентированности компании на развитие софтверных продуктов с открытым исходным кодом и ожидания существенного расширения API и более качественной поддержки разработчиков.

Архитектура предлагаемого решения приведена на рис. 7 и включает в себя:

- сервер приложения с расширенной функциональностью;
- база данных для хранения и обработки информации;
- клиентское мобильное приложение для Android.

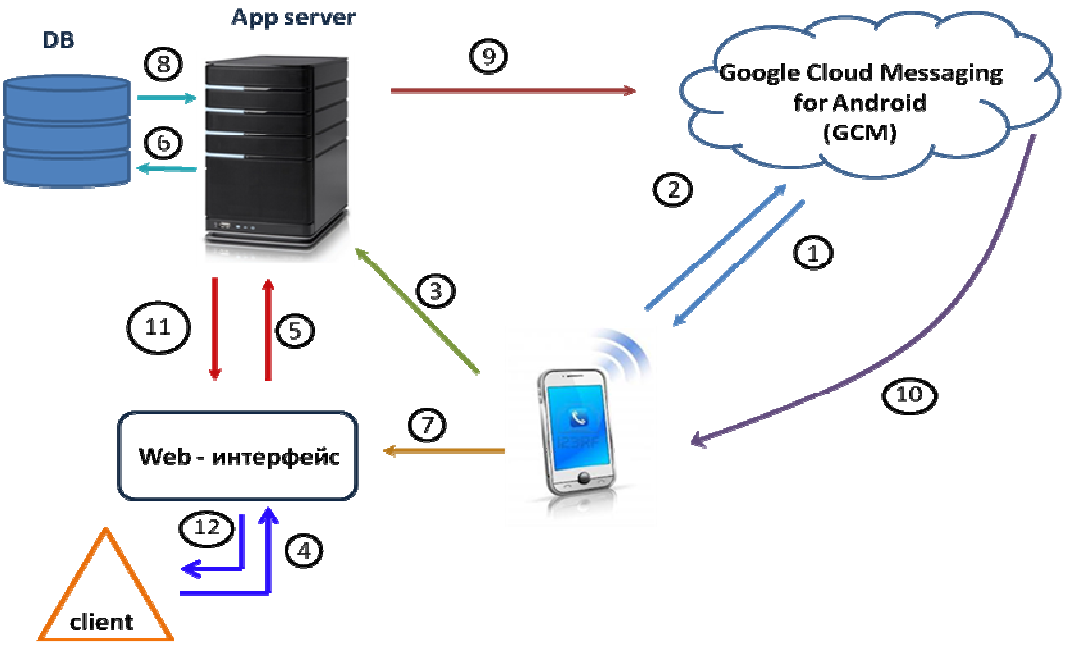

Рис.7. Архитектура проектируемой системы push-уведомлений

На первоначальном этапе разработки системы уведомлений были определены следующие основные требования:

1) формирование и отправка сообщения осуществляется с использованием web-интерфейса;
2) на мобильном устройстве по каждому сообщению можно идентифицировать отправителя;
3) отправленный текст должен представлять собой html-страницу, рассылка которой осуществляется в соответствии с определенным графиком;
4) вся информация по совершенным транзакциям должна хранится в реляционной базе данных;
5) легкая масштабируемость системы для новых проектов и заказчиков.

Алгоритм работы данной системы push-уведомлений с появлением нового функционала существенно расширился и помимо раннее рассмотренных этапов включает в себя следующие новые позиции:

- после регистрации мобильного устройства пользователя в системе ему предоставляется возможность определения наиболее интересных тем для подписки, а также планируемой к получению информации. В данном случае пользователь (рис. 4. цифра 7) через предоставляемый интерфейс приложения осуществляет все необходимые процедуры по подписке;

- клиент (заказчик системы рассылки) через предлагаемый веб-интерфейс осуществляет управление рассылкой (4). Непосредственно через веб-браузер происходит выбор тем, устанавливается график рассылки для каждого сообщения, вводится текст сообщения с использованием html-тегов;

- вся информация о совершенных действиях клиента и пользователя поступает на хранение в реляционную базу данных(5, 6).

2.2. Описание клиентской части

Клиентская часть приложения для пользователей мобильных устройств написана под операционную систему Android версии 2.3.3. (и выше) с использованием среды разработки Eclipse.

При работе с приложением необходимо пройти процедуру регистрации для предоставления идентификационной добровольной информации. Данный тип информации используется исключительно для возможности обращения при взаимодействии с пользователем и получении общей статистики по различным группам. К предлагаемым для заполнения полям относятся: имя, фамилия, пол, возраст, электронная почта. Регистрационная форма приведена на рис 8. Реализация процесса регистрации осуществляется в классе RegisterActivity.class

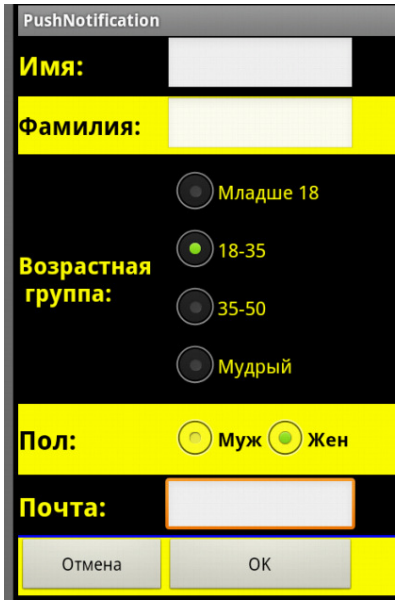

Рис.8. Регистрационная форма

Данная идентификационная информация сохраняется в базе данных «PushNotification» SQLite на мобильном устройстве, а также дублируется в централизованной базе данных Postgresql, используемой

сервером приложения для осуществления процесса отправки уведомлений.

Структура БД «PushNotification» приведена на рис.9. В таблицах «person» хранится регистрационная информация о пользователе, «message» - информация о входящих уведомлениях (компания отправитель, тема подписки, наименование сообщения и дата получения сообщения), «company» - связь между компанией и ее логотипом.

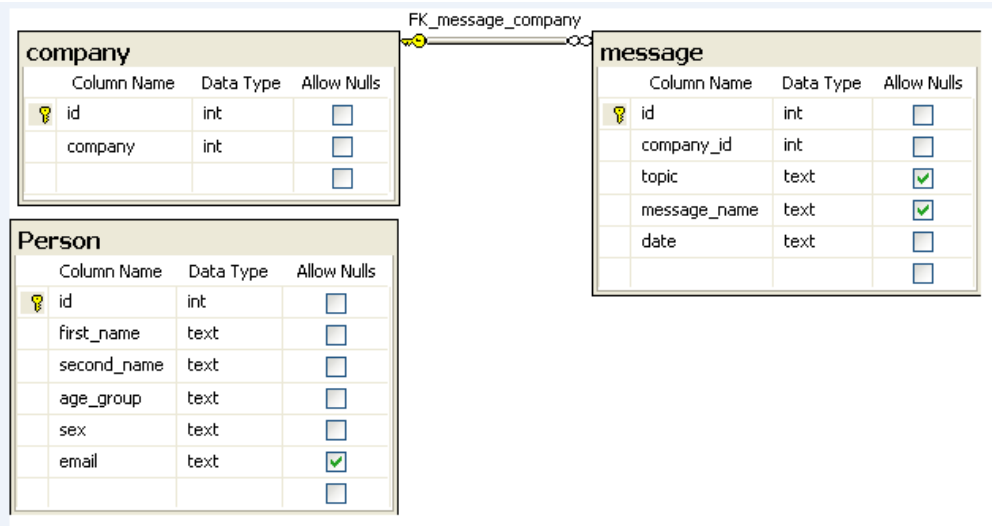

Рис.9. Диаграмма БД «PushNotification»

При работе с приложением пользователю предоставляется возможность совершать подписку на определенные темы, просматривать сообщения, осуществлять отказ от регистрации в системе.

Для удобства пользователя также разработано меню для возможности осуществления удаления всех ранее полученных сообщений, отказа от подписки в системе рассылки сообщений.

Приложение также позволяет осуществлять сбор анонимной статистики по возрасту, полу, времени открытия сообщения.

2.3. Описание серверной части

В качестве базовых технологий для разработки архитектуры серверной части системы push-уведомлений используется платформа JavaEE 6 (java enterprise edition). В текущей реализации планируется использовать стабильный и достаточно эффективный сервер приложений с открытым исходным кодом GlassFish 3.1.2, позволяющий реализовать все возможности данной платформы. Для общения с GCM-службой используется стандартный HTTPS протокол.

Ядро серверной части системы реализовано в качестве Java приложения с back-end базой данных Postgresql, в которой хранятся списки пользователей, клиентов, темы подписки и отправленные сообщения.

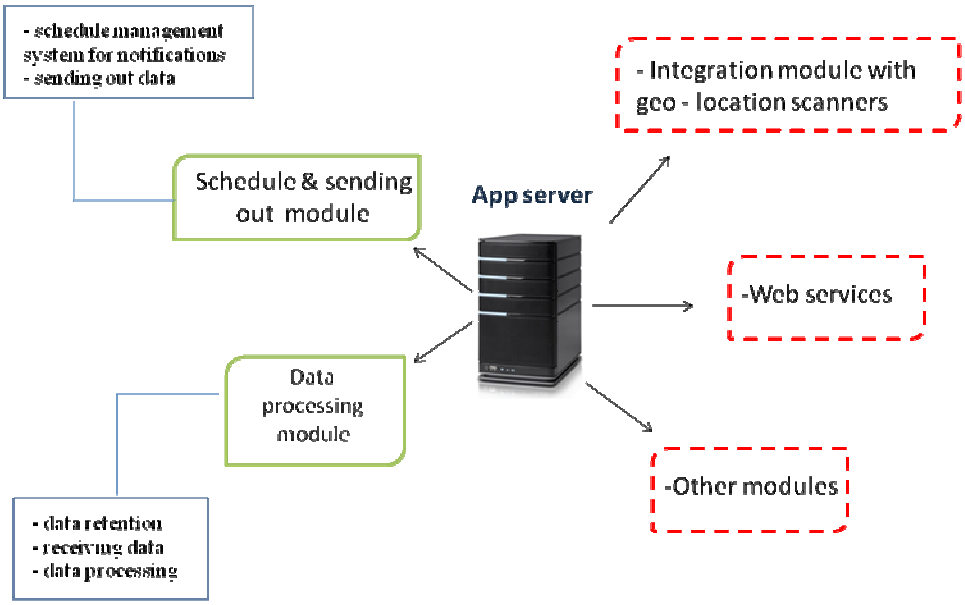

Рис 10. Архитектура серверной части

Общая схема архитектуры серверной части представлена на рис 10. Она включает в себя две реализованные подсистемы: модуль обработки данных, которые поступают на сервер, а также используются для формирования статистики, и модуль управления расписанием и рассылкой уведомлений.

Общий принцип работы системы

Для отправки сообщения приложение формирует POST запрос по адресу https://android.googleapis.com/gcm/send. В HTTP заголовке в обязательном порядке указывается поле авторизации, содержащее маркер отправителя, и тип контента (Content-Type: application/json). Поля тела сообщения приведены в таблице №2 [1].

Таблица №2

Наименование поля	Описание
Идентификационные номера зарегистрированных мобильных приложений (registration_ids)	Список (от 1 до 1000) регистрационных номеров приложений
Данные (data)	Текст сообщения
Задержка (delay_while_idle)	Сообщение отправляется только при условии, что устройство находится в активном режиме.
Время жизни (time_to_live)	Период, в течение которого сообщение может хранится в GCM службе, если устройство вне зоны действия сети (значение по умолчанию 4 недели).

По результатам отправки сообщений сервер приложений получает код ответа от GCM службы, позволяющий проанализировать сбои и ошибки в системе. Так, в случае успешной обработки сообщений в GCM, код ответа принимает значение 200, а в теле содержатся дополнительные информационные поля (см. таблицу №3) [1].

Таблица №3

Наименование поля	Описание
Уникальный номер группы сообщений (multicast_id)	Номер, присваиваемый GCM службой группе отправляемых сообщений.
Успех (success)	Количество успешно обработанных сообщений.
Неуспех (failure)	Количество сообщений, обработанных с ошибкой и неотправленных

В случае отклонения запроса на отправку уведомлений система генерирует коды 400, 401, 5xx. (подробная информация приведена в таблице №4) [1].

Таблица №4

Код ответа	Описание
400	Формат JSON не может быть правильно обработан, либо содержит невалидные поля.
401	Проблемы с аутентификацией сервера приложений в службах Google
5xx	Ошибки указывают на внутренние проблемы GCM службы.

Логика основной работы алгоритма по управлению поступающих сообщений на серверную часть представляет собой следующие этапы:

1) в соответствии с установленным расписанием EJB (enterprise java bean) производит выгрузку информационных данных в массив типа данных PriorityBlockingQueue по сообщениям, которые необходимо отправить в течение текущего дня. Целесообразность использования типа данных PriorityBlockingQueue обусловлена необходимостью обеспечить неблокирующий доступ к данным в режиме многопоточности;

2) далее обработка рассылки каждого сообщения осуществляется асинхронно с установлением расписания рассылки для пула потоков. Метод предполагает определение разницы во времени между последним отправленным уведомлением и текущим для установления временного лага рассылки уведомления;

3) при добавлении новых данных для рассылки также определяется время отправки, которое согласуется с текущим расписанием. Если уведомление необходимо отправить в день отличный от текущего, то данные о нем просто заносятся в базу данных. В противном случае уведомления, которые по времени должны быть отправлены позже текущей рассылки, заносятся в массив рассылки, а если раньше – асинхронно обрабатываются пулом потоков с новым для себя расписанием.

Помимо рассылки уведомлений заказчику также предоставлена возможность проводить некоторую аналитику по пользователям (количество, пол, возраст, активность), которая сохраняется в реляционной базе данных.

ГЛАВА III. Анализ применимости системы

3.1. Модели применения системы push-уведомлений

В настоящее время рост конкуренции на рынке заставляет большинство продавцов развивать различные программы лояльности среди своих клиентов. В большинстве случаев такой маркетинговый ход позволяет сохранить наиболее ценных клиентов.

Среди таких подходов активно развивается система глубокого анализа предпочтений клиента, основанная на отслеживании их поведения (например, в торговых сетях) для формирования уникального рыночного предложения. Все передвижения клиента фиксируются, на основе этого составляется карта маршрута со временем его прохождения. По полученным картам определяются: в каких отделах клиент задерживался, путь, по которому он проходит до необходимого товара, и прочие параметры.

Среди компаний, которые предоставляют возможность внедрить такую систему, являются Navizion, WalkBase, Cisco [4] [5] [6].

Основной принцип работы системы заключается в том, что сетевые узлы (сканеры) в фоновом режиме и абсолютно ненавязчиво определяют месторасположение мобильных устройств с включенным WiFi модулем. При этом нет необходимости устанавливать приложения. Все сканеры размещаются, как правило, на определенном расстоянии друг от друга с целью покрытия максимальной территории. Устройства автоматически конфигурируются под параметры сети и организуют сбор различной неконфиденциальной информации, которая затем передается заказчику.

Недостатком этой системы является невозможность персонификации клиента.

В совокупности с предложенной системой push-уведомлений заказчик существенно расширяет свои возможности по работе с клиентами за счет способности идентифицировать пользователя по MAC адресу WiFi-модуля в мобильном устройстве. При этом предполагается, что человек уже зарегистрирован в системе.

Таким образом, заказчик может осуществлять таргетированную рассылку пользователям, которые непосредственно осуществляют покупку товаров в определенные моменты времени.

В этом случае необходимо развивать графическую составляющую при работе с системой на клиентской и серверной частях приложений. В рамках такого подхода интересным решением является персонификация приложений с логотипом заказчика. В связи с этим важно рассмотреть общую архитектуру системы, которая позволила бы быстро (например, без дополнительной компиляции исходного кода) и самостоятельно силами клиента изменять настройки приложения и его графическую составляющую.

Некоммерческим вариантом может быть рассмотрена система, позволяющая отслеживать посещаемость и успеваемость ученика в школе или на отдельных специальных дополнительных мероприятиях. В данном случае возникает необходимость в двух клиентских приложениях для родителя и ученика, на которые будет приходить информация об изменениях в расписании, о домашних заданиях и т.д.

Следующая модель применения предполагает использование текущего варианта системы с технологией веб-сервисов, когда программа идентифицируется веб-адресом со стандартным интерфейсом.

Такой подход позволяет уже рассматривать разработанную систему как самостоятельное «облачное решение», которое берет полное управление по хранению, управлению всей необходимой информацией и рассылке уведомлений клиентам. В качестве веб-службы можно использовать RESTful, поскольку она отличается простотой и удобством, а также предоставляет возможность передачи данных непосредственно по HTTP.

3.2. Направления дальнейшего развития системы

Направления развития системы push-уведомлений в первую очередь связаны с моделями применения системы. Тем не менее, разработка многофункционального веб-интерфейса на серверной части позволит контролировать процесс системы уведомлений и сделает ее наглядной. Также она позволит визуализировать накопленную статистику для дальнейшего анализа.

Добавление поддержки веб-служб, например RESTful, позволит сторонним компаниям взаимодействовать с API серверной части и тем самым использовать мощности системы уведомлений и интегрировать их со своими приложениями и службами.

Дополнительная работа над брендированием мобильных приложений будет способствовать персонификации каждого приложения под заказчика, что благоприятно повлияет на ассоциативную связь его со своими клиентами.

Заключение

В настоящей работе был проведен анализ платформ для построения системы push-уведомлений основных крупных компаний-разработчиков операционных систем для мобильных устройств Google, Apple, BlackBerry. Предлагаемая ими архитектура легко масштабируема, позволяет использовать сторонние серверы и в качестве языка программирования может быть выбрана Java.

Также рассмотрены существующие коммерческие и open source аналоги, среди которых не представлены готовые системы, реализованные на Java. Среди проектов с открытым исходным кодом функциональность систем позволяет осуществлять только рассылку уведомлений.

Система push-уведомлений, предложенная в данной дипломной работе, позволяет вести учет по различным темам подписки в разрезе компаний, клиентов, а также осуществлять рассылку уведомлений с учетом расписания. Все уведомления во время рассылки обрабатываются асинхронно. Клиентское приложение под Android позволяет пользователям осуществлять регистрацию в системе, производить подписку к интересующим темам, непосредственно читать сообщения, поступающие в виде html-текста, а также формировать статистику для модуля аналитики.

Кроме того, в работе рассмотрены различные модели применения разработанной системы push-уведомлений в реальности, а также предложен ряд мероприятий по возможному дальнейшему усовершенствованию системы.

Список литературы

[1] Google Cloud Messaging for Android
http://developer.android.com/google/gcm/gs.html

[2] Using Local and Push Notifications On iOS and Mac OS X
Darryl Bleau, APNs Engineering Manager
http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Introduction.html

[3] Push Service
https://developer.blackberry.com/develop/platform_services/push_overview.html

[4] Indoor Triangulation System
http://www.navizon.com/its/whitepaper.pdf

[5] http://www.walkbase.com/

[6] Cisco Wireless Location Appliance
http://www.cisco.com/en/US/prod/collateral/wireless/ps5755/ps6301/ps6386/product_data_sheet0900aecd80293728.html

[7] Push Notifications
http://support.brightcove.com/en/app-cloud/docs/push-notifications

[8] Good Push does great things

http://urbanairship.com/products/push-messaging#analyze

[9] Mark L. Murphy (January, 2010). The Busy Coder's Guide to Android Development

[10] GlassFish Server Open Source Edition 3.1 Quick Start Guide. Oracle Corporation
http://glassfish.java.net/docs/3.1/quick-start-guide.pdf

[11] Владимир Павлов, Дмитрий Намиот “Анализ и Разработка Системы Push-уведомлений с Использованием Технологий Google”, International Journal of Open Information Technologies, 1(3), pp. 20-24.

[12] Dmitry Namiot. "Local Area Messaging for Smartphones." International Journal of Open Information Technologies 1.2 (2013): 8-11.