электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.4

Архитектура автоматизированной системы мониторинга серверов и сервисов компьютерной сети

Опрышко А. В., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Гапанюк Ю.Е., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
gapyu@bmstu.ru

Автоматизированная система мониторинга серверов и сервисов компьютерной сети (далее система) - предназначена для мониторинга и отслеживания статусов компьютерных систем: наблюдения, составление графиков производительности и контроля состояния серверов и сервисов, а также оповещения администратора о нештатных ситуациях.

Возможности:

- мониторинг сетевых служб (HTTP, ICMP)
- мониторинг состояния хостов в реальном времени (загрузка процессора, использование памяти и т. д.) в большинстве сетевых операционных систем
- составление графиков производительности в реальном времени (загрузка процессора, использование памяти)
- простая архитектура модулей расширений (плагинов) позволяет, используя любой скриптовый язык программирования (Python, Bash, Perl и другие), легко разрабатывать свои собственные способы мониторинга
- возможность создания правил для метрик, поступающих на сервер мониторинга, в случае невыполнения которых произойдет оповещение
- параллельная проверка служб
- отправка оповещений в случае возникновения проблем со службой или хостом (с помощью почты, смс и нотификация в реальном времени, если пользователь находится в системе)

Система состоит из 4 основных частей: сервис мониторинга, установленный на сервере, за которым ведется наблюдение, сервере мониторинга, сервер веб-приложения и сервер уведомлений. На рисунке 1 представлена обобщенная архитектура системы.

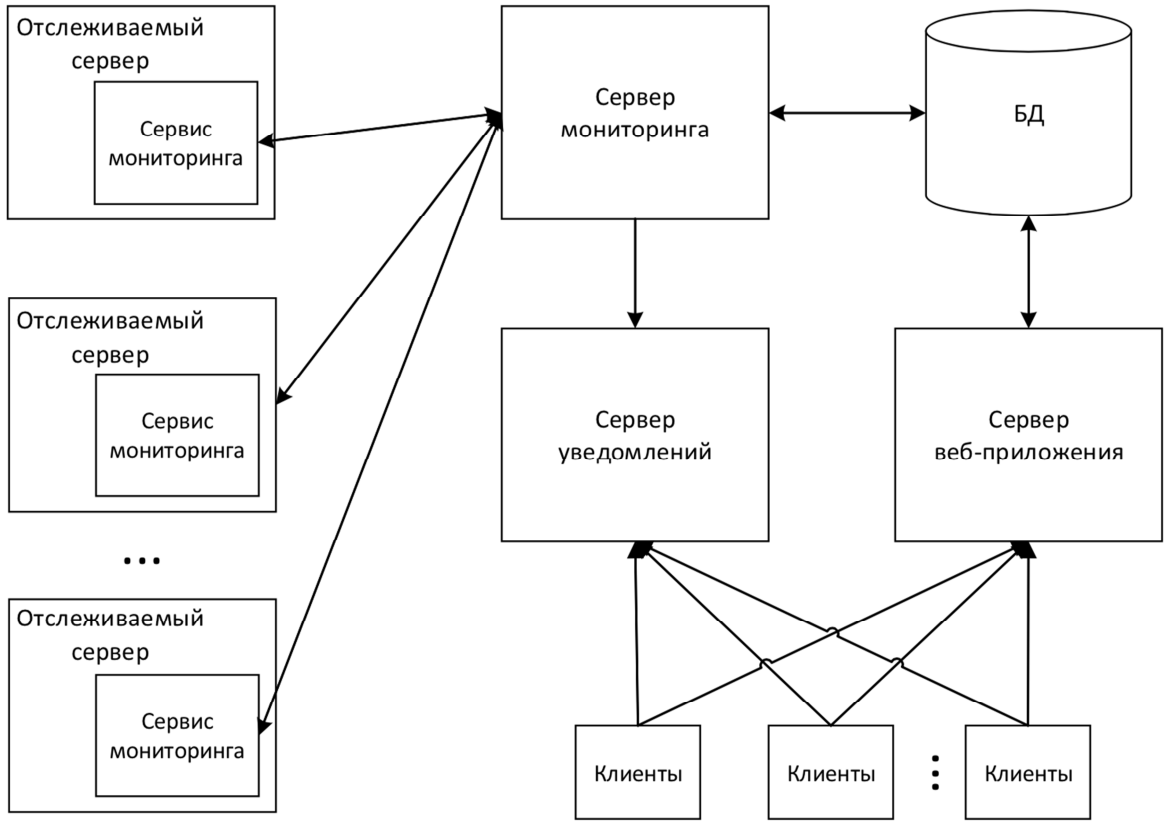

Рис. 1. Обобщенная архитектура системы

Сервис мониторинга. Сервис запускается в фоновом режиме на сервере, за которым ведется наблюдение. Сервис написан на языке C++ с использованием библиотек Boost и std.

Boost - собрание библиотек классов, использующих функциональность языка C++ и предоставляющих удобный, кроссплатформенный, высокоуровневый интерфейс для лаконичного кодирования различных повседневных подзадач программирования (работа с данными, алгоритмами, файлами, потоками и т.п.).

STD (Стандартная Библиотека) - коллекцию классов и функций, написанных на базовом языке. STD поддерживает несколько основных контейнеров, функций для работы с этими контейнерами, объектов-функции, основных типов строк и потоков (включая

интерактивный и файловый ввод-вывод), поддержку некоторых языковых особенностей и часто используемые функции.

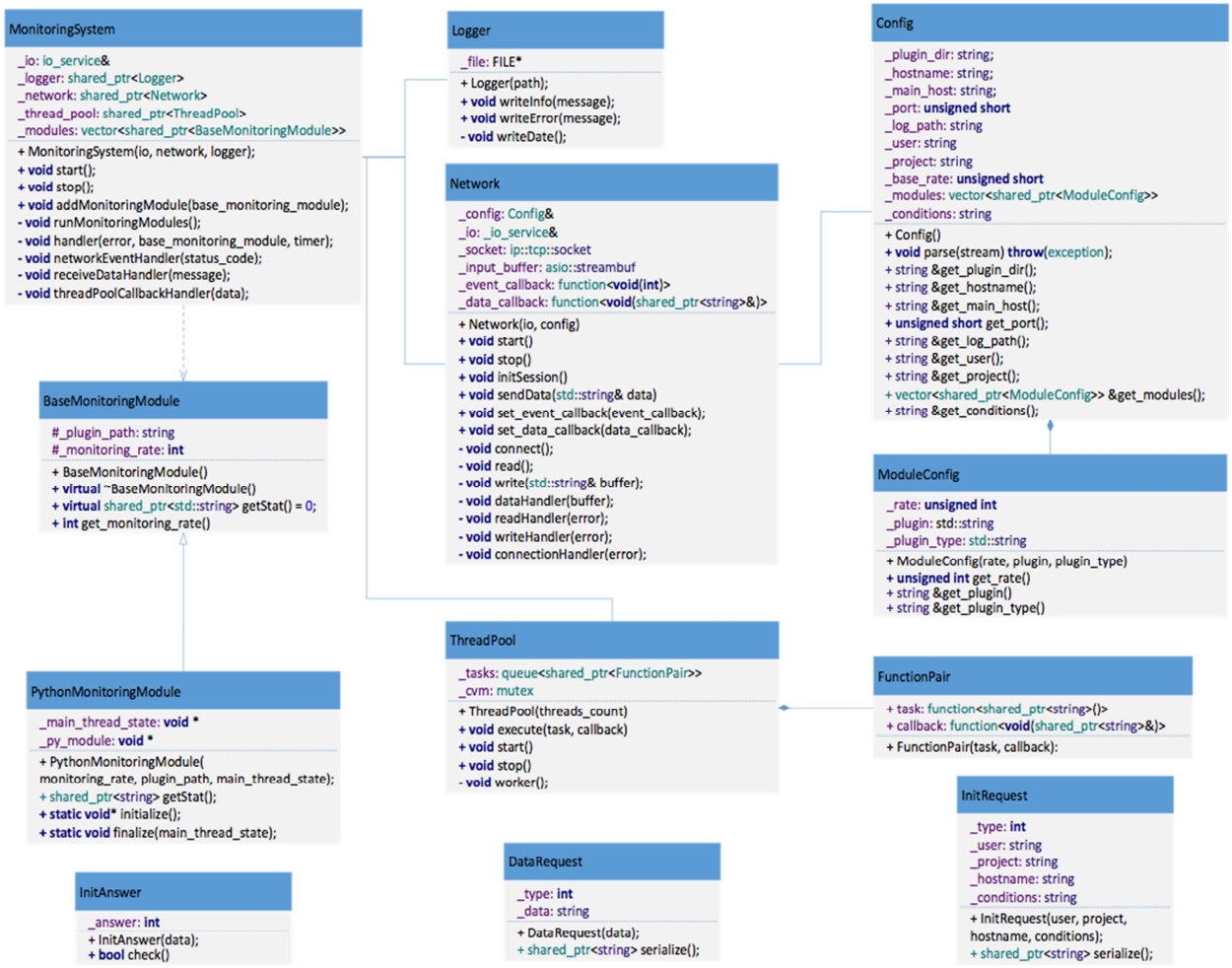

Рис. 2. UML диаграмма

Рассмотрим диаграмму классов, представленную на рисунке 2. Корнем системы является класс MonitoringSystem.

Основные функции MonitoringSystem:

- инициализация ThreadPool для выполнения длительных задач
- инициализация сетевого взаимодействия с сервером
- получение метрик по расписанию
- отправка метрик на сервер
- обработка сообщений от сервера

Сетевое взаимодействие реализовано, по протоколу TCP, в классе Network.

Основные функции Network:

- установление и разъединение TCP соединения

- установление логического соединения с сервером
- отправка данных по сети серверу
- получение данных от сервера

Непосредственно получение метрик происходит в классах BaseMonitoringModule и PythonMonitoringModule. BaseMonitoringModule является абстрактным классом с чистой виртуальным методом getStat(). PythonMonitoringModule наследуется от BaseMonitoringModule, в котором реализуется getStat, в которой происходит получение метрик, посредством выполнения Python скриптов. Следовательно, для добавления нового модуля мониторинга, необходимо наследоваться от BaseMonitoringModule и реализовать метод getStat, для конкретного скриптового языка. Такая архитектура позволяет гибко добавлять новые модули мониторинга.

Конфигурация системы мониторинга храниться в JSON (текстовый формат хранения и обмена данными) файле и обрабатывается классом Config и ModuleConfig.

Пример конфигурации:

```
{
    "plugin_dir": "/path_to_plugins/",          #путь к плагинам
    "hostname": "hostname",                     #имя локального хоста
    "main_host": "127.0.0.1",                   # ip адрес сервера мониторинга
    "port": 8888,                               # порт сервера мониторинга
    "log_path": "/path_to_log/event.log",       # путь к log файлу
    "user": "test@mail.ru",                     # email пользователя в системе
    "project": "title",                         # имя проекта
    "base_rate": 23,                            # период времени между 2мя отправками метрик (c)
    "modules":[{
            "rate":1,
            "plugin":"disk.py",                 # имя плагина
            "plugin_type":"py"                  # тип плагина
    }],
    "conditions": {                             # условия при при которых произойдет уведомление
      "folder": {
        "metric": {
          "type": "m",
          "info": "#arg# > 10",                 # условие отправки info сообщения
          "warning": "#arg# > 27",              # условие отправки warning сообщения
          "dangerous": "#arg# > 30"             # условие отправки dangerous сообщения
        }
      }
    }
}
```

Управление множеством потоков для выполнения задач происходит в классе ThreadPool. На вход поступает задача, если нет свободных обработчиков, то задача попадает в очередь. Как только какой-то обработчик из ThreadPool освободился он

выполняет эту задачу и возвращает результат. На рисунке 3 представлена схема работы ThreadPool.

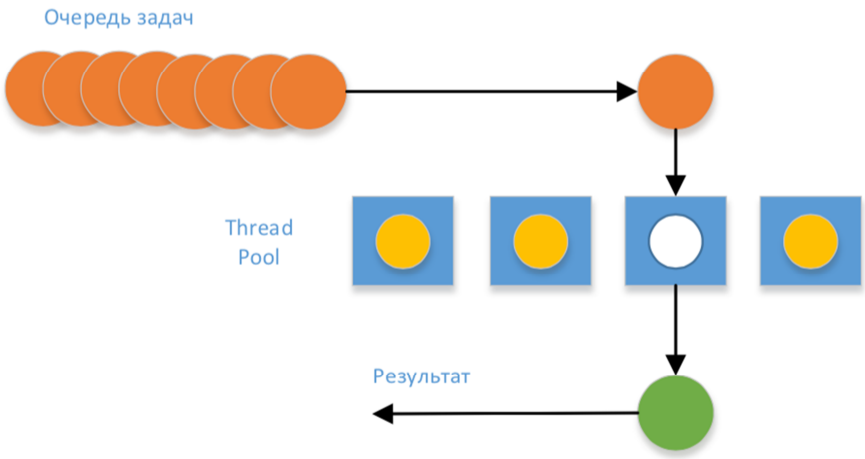

Рис. 3. Схема работы ThreadPool

Для взаимодействия с сервером мониторинга необходимо установления логического соединеия. Для этого необходимо отправить запрос, описанный классом InitRequest. Для отправки данных на сервер мониторинга необходимо отправить запрос, описанный классом DataRequest.

Сервер мониторинга. Может находится в трех режимах работы:

1. Инициализация логического соединения (алгорит представлен на рисунке 4)
2. Получения метрик от клиента. (алгоритм представлен на рисунке 5)
3. Завершение соединения

Инициализация логического соединения. Для организации клиент-серверного взаимодействия необходимо получить идентификатор хоста в системе и сохранить его в текущую сессию.

Получения метрик от клиента. Для сохранения метрик на сервере необходимо реализовать хранение древовидных данных в реляционной базе данных. Для оптимизации доступа к данным выполним денормализацию и для каждой метрики помимо id родительской метрики будем хранить полный путь к ней (даталогическая модель БД представлена на рисунке 6).

Завершение соединения - необходимо чтобы клиент закрыл соединение.

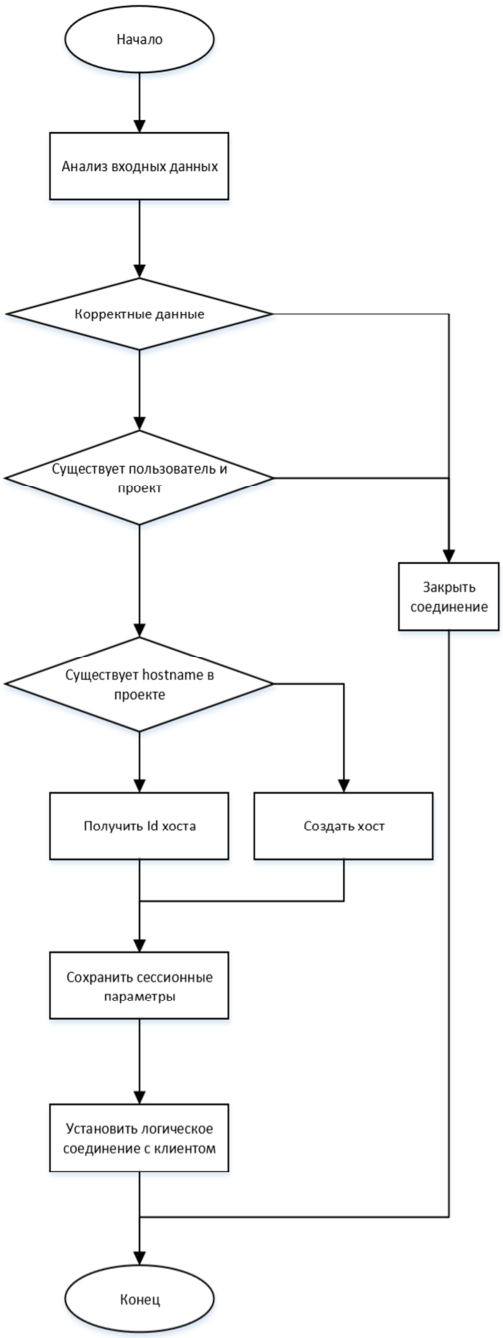

Рис. 4. Инициализация логического соединения

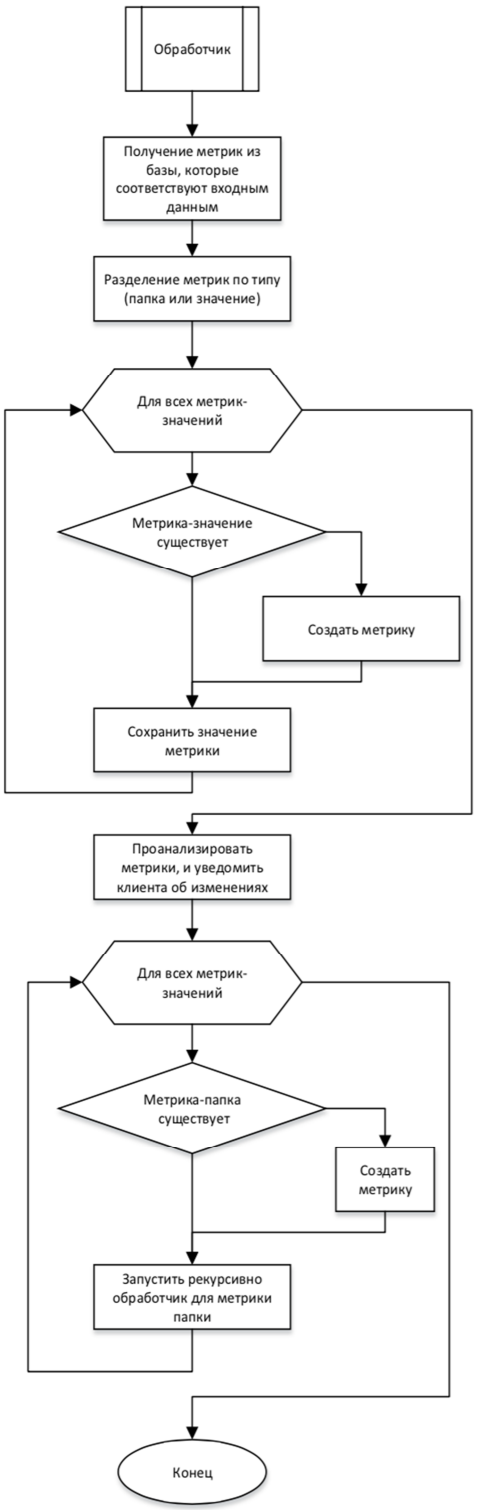

Рис. 5. Получения метрик от клиента

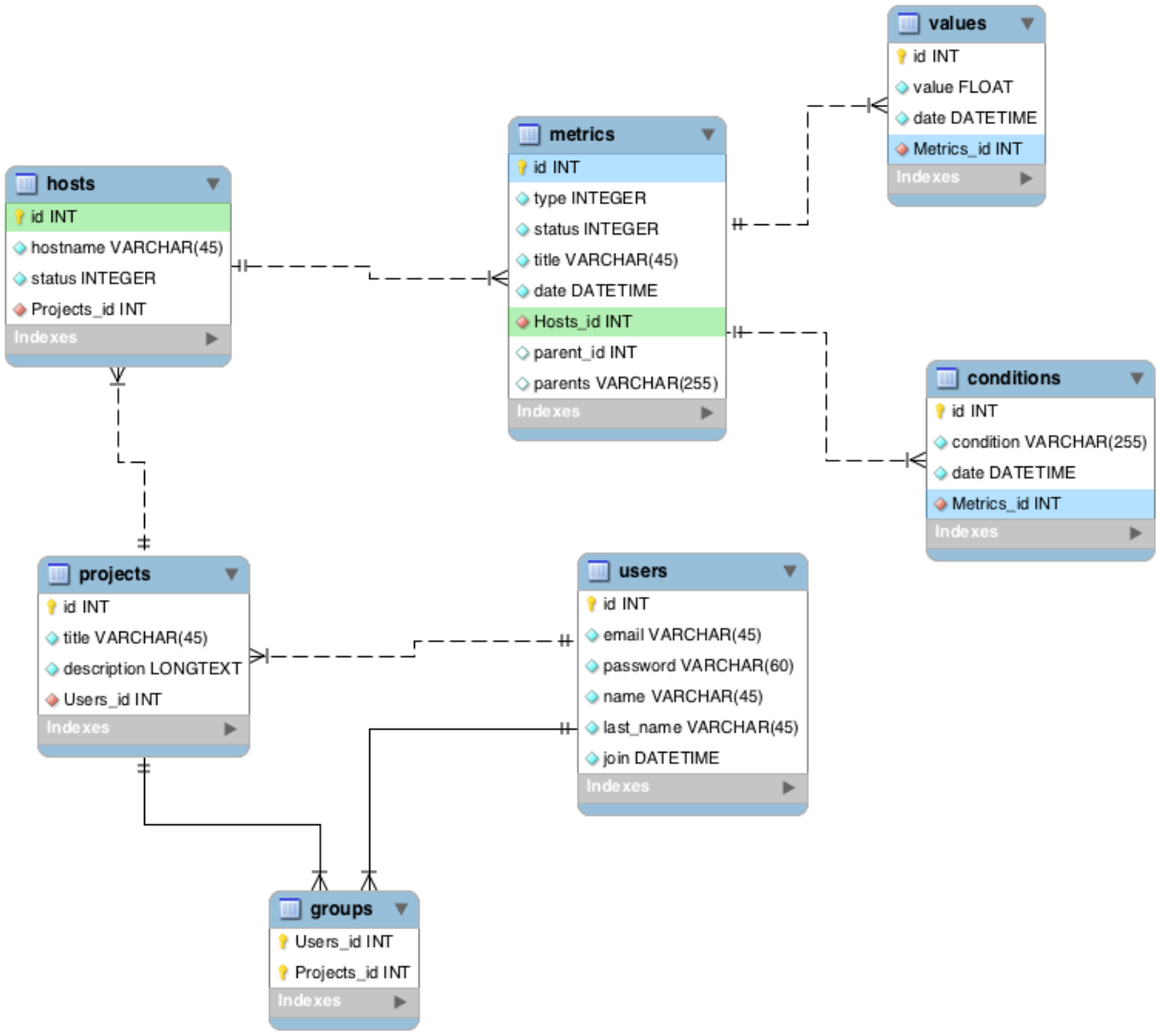

Рис. 6. Даталогическая модель БД

Сервер веб-приложения. Сервер веб-приложения построен на основе фреймворка Tornado. Tornado - расширяемый, неблокирующий веб-сервер и фреймворк, написанный на Python. Сервер веб-приложения представляет собой REST API.

Клиентское приложение построено на основе фреймворка Backbone и с архитектурой Model-view-controller (MVC). MVC - схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные.

Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

- Model - данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать
- View - отвечает за отображение информации (визуализацию)
- Controller - обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции

Сервер уведомлений. Сервер уведомлений позволяет получать уведомления в реальном времени о любых событиях, произошедших на сервере. Сервер построен на основе фреймворка Tornado и технологии кросбраузерного двунаправленного взаимодействия с клиентом sockjs.

Для получении уведомления о конкретном событие клиент должен подписаться на него. Для отправления уведомления сервер мониторинга использует REST API сервера уведомлений. При получении запроса, сервер определяет тип события, получает список подписанных пользователей и рассылает уведомления.

Выводы. Применение, описанной в статье архитектуры, позволяет организовать гибкую систему мониторинга и отслеживания статусов серверов компьютерной сети в любой инфраструктуре. Также система, построенная по данной архитектуре, является полностью автономной и не требует установки дополнительных сервисов.

Список литературы

1. Кнут Д. Искусство программирования. В 4 т. Т. 1. Основные алгоритмы. М.: Вильямс, 2006. 720 с. [Donald E. Knuth. The Art of Computer Programming, vol. 1. Fundamental Algorithms, Third Edition. Addison-Wesley, 1997. 650 p.].
2. Lutz M. Learning Python, 5th Edition. Sebastopol: O'Reilly Media, 2013. 1600 p.
3. Официальная документация Tornado. Режим доступа: http://tornado.readthedocs.org/ (дата обращения 28.04.2015).
4. Официальная документация Backbone.js. Режим доступа: http://backbonejs.org/ (дата обращения 28.04.2015).
5. Gapanyuk Yu., Lakomkin E., Ionkin S., Davtyan M. MVC WEB Framework based on eXist application server and XRX architecture // 7th Spring Researchers Colloquium on Databases and Information Systems (Moscow, 02-03 June 2011). P. 19-25.