ИПМ им.М.В.Келдыша РАН

Абрау-2018 • Труды конференции

А.Ю. Шевчук, Ю.В. Шевчук

Веб-интерфейс системы мониторинга Botikmon

Рекомендуемая форма библиографической ссылки

Шевчук А.Ю., Шевчук Ю.В. Веб-интерфейс системы мониторинга Botikmon // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17-22 сентября 2018 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2018. — С. 499-510. — URL: http://keldysh.ru/abrau/2018/theses/24.pdf doi:10.20948/abrau-2018-24

Веб-интерфейс системы мониторинга Botikmon

А.Ю. Шевчук, Ю.В. Шевчук

ИПС имени А. К. Айламазяна РАН, ИЦМС

Аннотация: В статье рассматривается веб-интерфейс общецелевой системы мониторинга Botikmon. Система Botikmon сочетает в себе функциональность брокера сообщений и архива сенсорных данных; веб-интерфейс обеспечивает визуализацию данных из архива и редактирование метаданных. Веб-интерфейс поддерживает как формирование графиков "на лету" с поиском нужных данных в архиве, так и отображение заранее подготовленных страниц с графиками и сопроводительным текстом. При оформлении графиков поддерживается возможность сочетания на одном графике данных с разными порядками величин за счёт использования двух осей ординат. Поддерживается визуализация не только численных данных, но и данных перечислимых типов в виде цветовых диаграмм. В статье описаны общие принципы организации веб-интерфейса, создания, группировки и описания визуализаций, а также модель пользовательского взаимодействия с визуализациями при выборе интересующего пользователя участка данных. Описан механизм "отражений", позволяющий использовать ранее подготовленные страницы с графиками для отображения других данных без затрат усилий на оформление новых страниц. Приводится обзор использованных в разработке технологий с краткой аргументацией в пользу выбора той или иной технологии.

Ключевые слова: сенсорные данные, временные ряды, визуализация, веб-интерфейс

Web-interface of the Botikmon monitoring system

A. Y. Shevchuk, Y. V. Shevchuk

PSI RAS, RCMS

Abstract: The following paper discusses the web interface for the Botikmon monitoring system. Botikmon is a general purpose monitoring system that acts both as a broker and as a data archive. The web interface provides data visualization and metadata editing capabilities for the system. The web interface allows for creating graphs on the go by searching for sensors in the archive as well as displaying previously saved pages, which consist of previously created graphs and descriptive text. Graphs support two ordinate axes which allows for displaying several sets of data of different orders of magnitude on the same graph. Aside from displaying numerical data using graphs, data of enumerated types can be displayed using color diagrams. The paper discusses the overall structure of the interface, the ways of creating, grouping together,

and describing graphs, as well as the user interaction model, which allows the user to display a time period of interest on the graph. The mechanism of "reflections" is described, which allows for reusing previously created pages and graphs for displaying similar data without an effort of creating new pages and graphs. Finally, the technologies used in development are listed, accompanied by the rationale for the choice of the specific technologies.

Keywords: sensor data, time series, visualization, web-interface

1. Введение

Botikmon представляет собой общецелевую систему мониторинга, сочетающую в себе функции брокера и архива сенсорных данных, ориентированную на большое количество источников (датчиков), порядка 10^6. Система принимает данные от многочисленных источников, транслирует их клиентам-подписчикам в реальном времени и сохраняет в архиве для ретроспективного анализа в будущем.

Наряду с бурным развитием автоматических средств анализа сенсорных данных, не теряет актуальности и проверенный временем метод пристального квалифицированного взгляда аналитика. Таким образом, существует потребность в удобной для аналитика визуализации сенсорных данных. Описываемый в данной статье веб-интерфейс системы как раз представляет собой инструмент ретроспективного анализа.

Существует множество систем визуализации временных рядов как для узкоспециальных задач и конкретных систем хранения, например Kibana [12] и Datadog [13] (мониторинг облачных приложений), так и общего назначения, например Graphana [14]. Помимо собственно визуализации данных разных типов, функциональность таких систем часто включает объединение графиков в приборные панели (dashboards), уведомление пользователей о специальных случаях, инструменты управления наблюдаемыми системами и даже некоторые функции социальных сетей. Веб-интерфейс Botikmon тоже поддерживает создание страниц с заранее подготовленным набором графиков (аналог приборной панели), но даже большее внимание уделяет легкости формирования графиков «на лету» с поиском нужных данных в архиве.

Архитектура системы в общих чертах представлена на рис. 1.

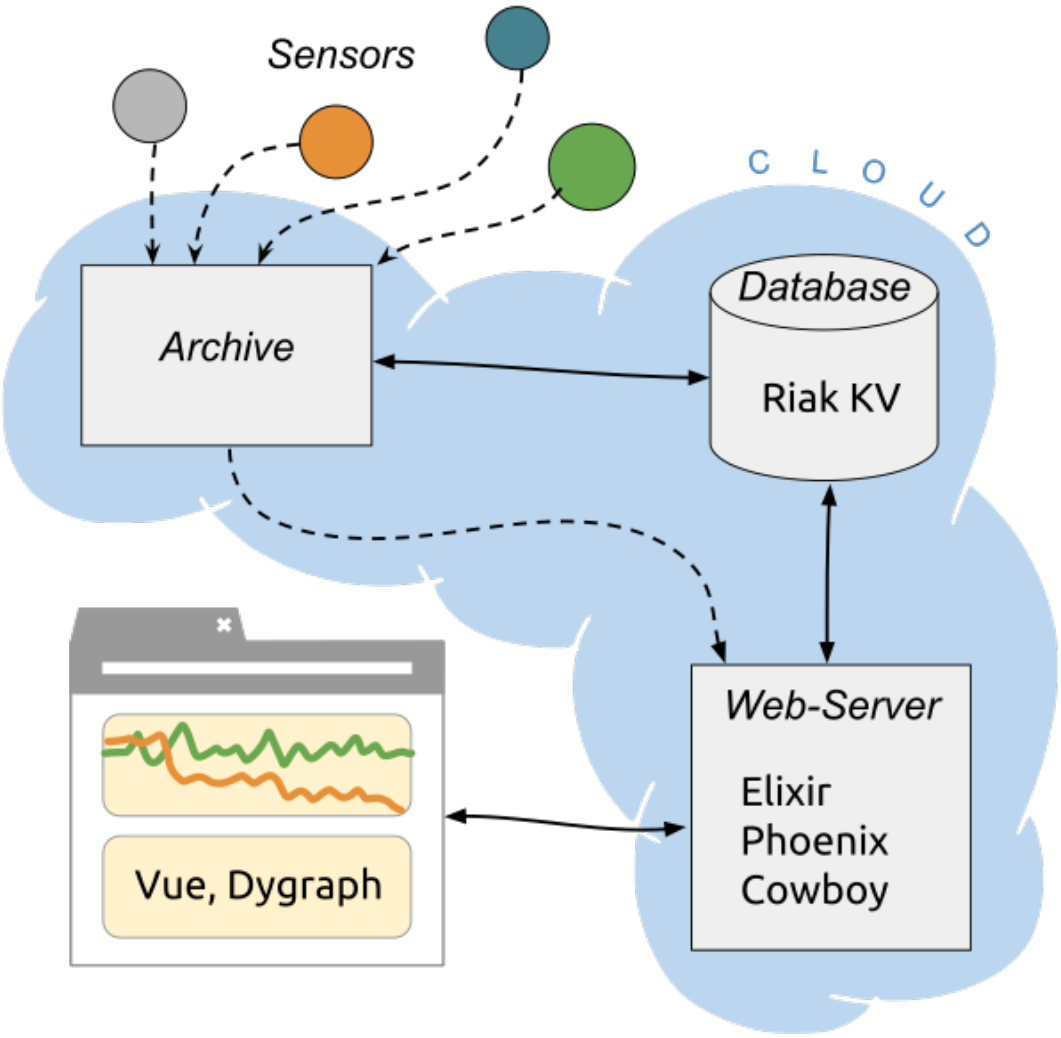

Рис. 1. Обзор архитектуры системы мониторинга

2. Типы сенсорных данных и способы их визуализации

В данный момент в рассматриваемых сенсорных сетях случаются данные трёх типов, если рассматривать с позиции визуализации:

1. Счетчики. Например, количество сетевых пакетов, принятых на порту коммутатора, или количество секунд с момента последнего включения устройства (uptime).
2. Результаты измерений. Например, показания датчиков температуры, влажности воздуха, напряжения в сети.
3. Категории состояния. Например, показания датчиков движения (движение либо есть, либо его нет), открытия двери (дверь либо открыта, либо закрыта) или качественная характеристика напряжения в сети (нормальное, низкое, высокое и так далее).

Важной особенностью является объем сенсорных данных. Данные от каждого датчика могут приходить с частотой до нескольких раз в минуту и накапливаться в течение нескольких лет. При этом пользователь может быть заинтересован как в общей картине происходящего в течение месяцев или даже лет, так и в ежеминутных деталях на уровне индивидуальных показаний датчика. Очевидно, отображение на графике индивидуальных показаний датчика в течение длительных интервалов времени невозможно.

В таких случаях данные группируются по корзинам (например, по минутам, часам или дням) и агрегируются способом, специфичным для типа данных. Агрегация осуществляется на веб-сервере и в системе архивирования, таким образом радикально уменьшая объем данных, передаваемых клиенту для отображения. Например, агрегированные по дням данные в течение года составляют всего 365 точек, тогда как исходные данные, при условии поступления с частотой раз в 60 секунд, составляют более полумиллиона точек.

Данные первых двух типов можно отображать в виде графика, где время снятия показания (или, в случае агрегированных данных, начало временного интервала корзины) откладывается по оси абсцисс, а значение, полученное от датчика (или агрегированное значение), по оси ординат. При агрегации для каждой корзины вычисляется среднее арифметическое, а также максимальное и минимальное значения, которые отображаются в виде линии и "коридора" вокруг линии, соответственно.

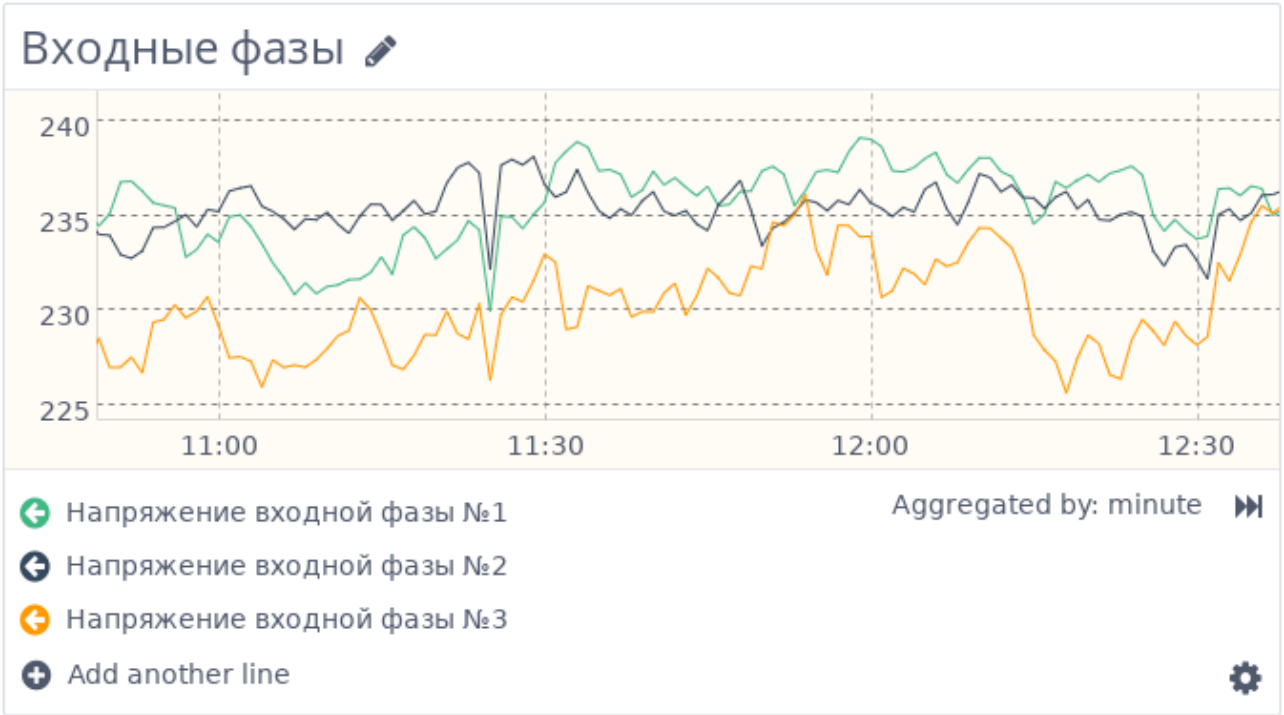

Рис. 2. Пример отображения сенсорных данных графиком

Третий тип данных (категории состояния) визуализируется несколько более оригинальным способом. Чтобы отобразить одну корзину агрегации,

внутри корзины вычисляется процент времени интервала, принадлежащего каждой категории; корзина отображается в виде разноцветного полосатого столбца, где полоска каждого цвета занимает соответствующий ассоциированной с этим цветом категории процент высоты столбца.

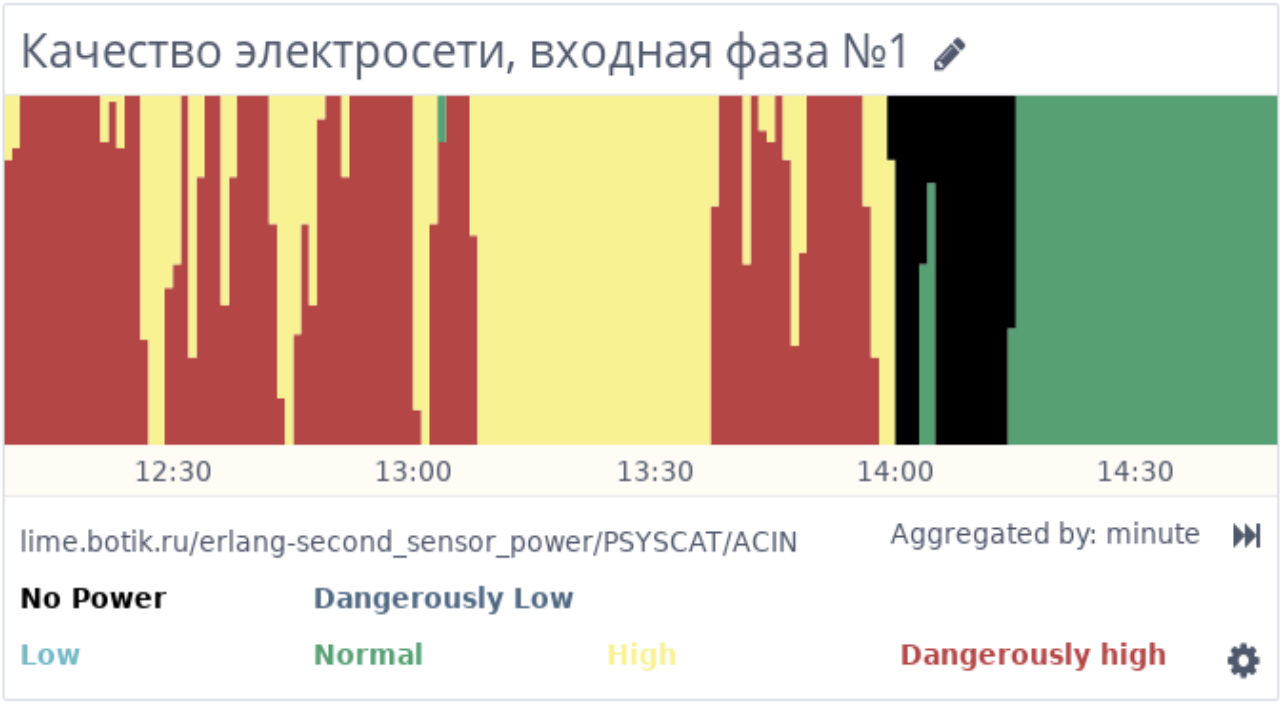

Рис. 3. Пример отображения агрегированных данных категорий состояния

Этот способ визуализации подходит также и для неагрегированных данных: в этом случае каждый столбец просто оказывается на 100% занят какой-то одной категорией и описывает интервал времени от одного показания датчика до другого вместо равных интервалов в случае агрегированных столбцов.

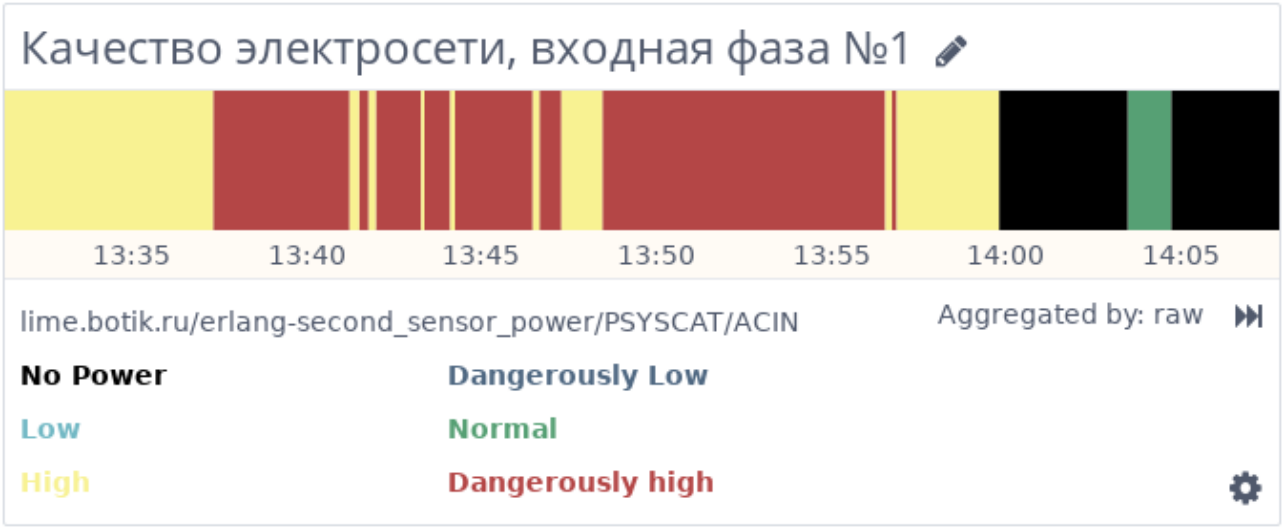

Рис. 4. Пример отображения неагрегированных данных категорий состояния

3. Элементы пользовательского интерфейса

Сенсорная сеть может обслуживать значительное, порядка 10^6, количество датчиков. Так, например, система мониторинга телекоммуникационного оборудования регионального Интернет-провайдера, первого из приложений описываемой системы, обслуживает около 300000 датчиков. При этом некоторые являются так или иначе связанными по смыслу: например, могут принадлежать одному коммутатору, одному порту коммутатора или одному и тому же показателю на разных портах.

В связи с этим, интерфейс должен обеспечить возможность визуализации любого отдельно взятого датчика "по требованию" (on demand), но также и предоставить средства визуализации и описательные средства для групп связанных датчиков.

На самом общем уровне, интерфейс состоит из набора именованных, создаваемых пользователями страниц, каждая из которых состоит из набора "виджетов". Пользователь может создавать, удалять и перемещать виджеты на странице друг относительно друга. В данный момент реализовано три типа виджетов:

1. График. На графике могут отображаться несколько линий для датчиков первого и второго типов.
2. Диаграмма категорий состояния. Отображает один датчик третьего типа.
3. Текстовый блок. Позволяет вставить произвольный текст на страницу. Поддержано форматирование текста с использованием Markdown.

При добавлении линий на график или виджетов на страницу, доступен поиск по датчикам сенсорной сети с поддержкой регулярных выражений, а также по уже существующим виджетам, присутствующим в системе.

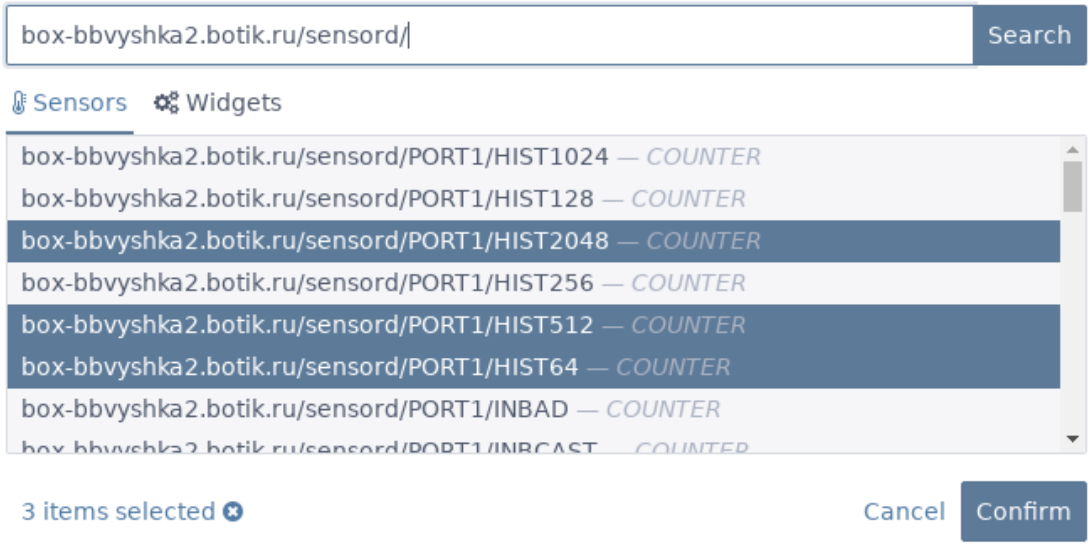

Рис. 5. Интерфейс поиска

Модель взаимодействия с графиком похожа на таковую в приложениях типа Yandex Maps, Google Earth и подобных. Пользователь может "перетаскивать" график "лапкой" и менять уровень приближения колёсиком мыши. При этом новые данные запрашиваются автоматически в фоновом режиме, не нарушая интерактивности графика. Уровень агрегации данных также выбирается автоматически так, чтобы обеспечить достаточный, но не бессмысленно подробный уровень детализации во избежание излишнего и потенциально чрезмерного объема передаваемых по сети данных.

В целом, высокая степень интерактивности интерфейса предоставляет удобные средства для быстрого поиска и отображения нужных данных, не требуя какой-либо предварительной конфигурации со стороны пользователя.

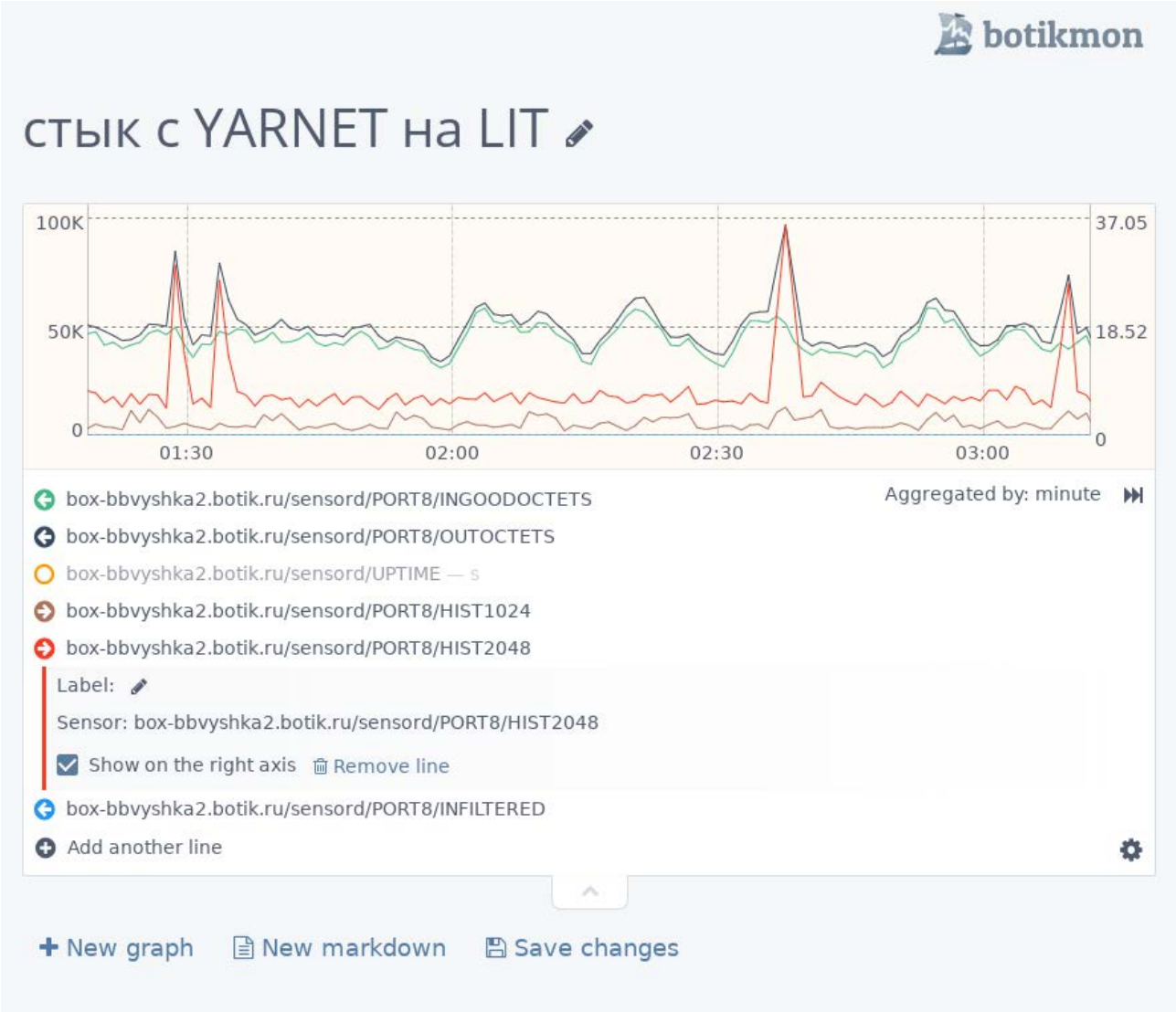

Рис. 6. Пример страницы с одним графиком

4. Механизм отражений

Механизм отражений позволяет использовать однажды созданную для одного устройства страницу для визуализации другого устройства того же типа. В языках программирования подобные приёмы называются "повторным использованием кода" (code reuse), здесь же повторно используются конфигурации страниц и виджетов.

Сенсорная сеть может содержать множество однотипных устройств, содержащих одинаковый набор датчиков. Например, несколько коммутаторов могут быть устройствами одной и той же модели. Идентификаторы датчиков однотипных устройств будут отличаться только идентификатором самого устройства. Например,

elt-trud1office/snmp-switch/te1/0/1/ifInOctets
elt-trud1office2/snmp-switch/te1/0/1/ifInOctets

Чтобы "отразить" страницу, нужно передать в параметрах HTTP-запроса список подстрок для замены в конфигурации страницы (и виджетов, присутствующих на ней). Например:

?p=elt-trud1office&r=elt-trud1office2

По итогам такого запроса все вхождения elt-trud1office на странице будут заменены на elt-trud1office2, в том числе в идентификаторах датчиков. Обе части замены здесь — это идентификаторы устройств (хотя механизм отражений к этому безразличен, поскольку оперирует непосредственно над текстом конфигурации). Если страница изначально была создана для устройства elt-trud1office, то после подстановки ("в отражении") станет описывать другой экземпляр такого же устройства, elt-trud1office2.

Такой подход замечателен, в первую очередь, отсутствием дополнительных нагромождений вроде специального языка шаблонов [2]. В момент создания страницы, описывающей первое устройство, пользователю даже не приходится думать о перспективах повторного использования.

Отражения работают также для отдельных виджетов и могут присутствовать даже на неотраженной странице.

Отражения не имеют собственной сохраняемой в базе данных конфигурации и поэтому их нельзя редактировать. Вместо этого можно отредактировать конфигурацию прообраза, и изменения унаследуются отражением автоматически при следующем обращении.

5. Библиотеки и технологии

В первую очередь, стоит упомянуть, что система архивирования сенсорных данных [1], к которой клиент (транзитом через веб-сервер) обращается за данными датчиков, написана на языке Erlang [10], и для распределенного хранения использует базу данных Riak KV [7] (которая тоже написана преимущественно на Erlang).

Erlang предназначен для написания распределенных, параллельных программ, в частности, серверных приложений. Веб-сервер является как раз такой программой, особенно если в будущем от одного сервера придется переходить к распределенной архитектуре. Таким образом, есть смысл серверную часть веб-сервиса также реализовывать на Erlang.

Хотя существующие в экосистеме Erlang веб-фреймворки не получили большого распространения, в последние годы набирает популярность фреймворк Phoenix [8], написанный на языке Elixir [9]. Elixir компилируется под виртуальную машину Erlang VM и хорошо интегрируется с программами на Erlang, вплоть до совместного исполнения в составе одной программы. Phoenix работает на базе веб-сервера Cowboy, также написанного на Erlang, и считается одним из самых производительных фреймворков сегодня [3].

В качестве базы данных используется тот же Riak KV. Объекты, сохраняемые в Riak, сериализуются в JSON, так что их можно передавать клиенту без предварительной десериализации.

На стороне клиента было решено использовать популярную сегодня архитектуру Single Page Application (SPA), то есть генерировать HTML-код страницы уже на клиенте с помощью JavaScript. Это выглядит оправданным ввиду высокой интерактивности интерфейса: клиентскому коду так или иначе придется работать с вёрсткой страницы, и дублировать эту функциональность на сервере было бы излишним. Многие похожие приложения, например Yandex.Metrika и Google Analytics, являются SPA.

Сегодня существует множество фреймворков, так называемых Client-side MVC, для удобной работы с SPA. Мы выбрали Vue.js [6], за хорошую производительность в тестах [4], опрятный API и отличную документацию.

В качестве CSS-фреймворка используется Spectre [5], как более минималистичная и менее узнаваемая в умолчательном стиле альтернатива Bootstrap.

Наконец, для построения графиков используется библиотека Dygraph [11]. Dygraph использует canvas для отрисовки и за счет этого имеет хорошую производительность даже при большом количестве точек. Dygraph умеет рисовать только линии, но при этом имеет исключительно гибкую конфигурацию, позволяя пользователю задавать собственную реализацию многих внутренних функций. Кроме прочего, это единственная библиотека из найденных, позволяющая программно устанавливать временной интервал отображения вне зависимости от загруженных данных, что необходимо при нашей модели взаимодействия с графиком.

6. Заключение

Веб-интерфейс системы мониторинга Botikmon предоставляет аналитику набор средств визуализации, организации и описания сенсорных данных. Интерфейс разработан минималистичным, по возможности интуитивно понятным и не требующим от пользователя предварительного изучения и специальных знаний (например, знания языка запросов к архиву сенсорных данных).

Высокая степень интерактивности интерфейса способствует динамичному рабочему процессу, позволяя аналитику быстро найти в системе и визуализировать интересующие его датчики, описать и прокомментировать полученную визуализацию и сохранить удачную конфигурацию приборной панели для последующего быстрого обращения.

Интерактивные графики с динамической подгрузкой данных и автоматическим выбором уровня детализации позволяют пользователю легко и наглядно выбрать интересующий его временной интервал и отобразить данные

с максимальной осмысленной степенью подробности, не передавая при этом избыточных данных по сети.

Предусмотрен механизм “отражений” для визуализации множества однотипных объектов путем повторного использования конфигурации однажды созданных приборных панелей и графиков, описывающих один такой объект. Таким образом, отображение множества объектов оказывается не сложнее, чем отображение одного единственного объекта, и требует минимальной дополнительной конфигурации.

Интерфейс введен в эксплуатацию и показал свою эффективность на практике. По мере расширения области применения системы мониторинга Botikmon, ожидается увеличение состава виджетов и способов визуализации, что легко вписывается в архитектуру интерфейса.

Литература

1. Живчикова Н.С., Шевчук Ю.В. Масштабируемая распределенная система архивирования сенсорных данных // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2017. — С. 157-169. — URL: http://keldysh.ru/abrau/2017/35.pdf doi:10.20948/abrau-2017-35
2. М. В. Стоцкий, Ю. В. Шевчук, “Шаблоны в модуле визуализации web-интерфейса системы мониторинга BotikMon”, Программные системы: теория и приложения, 5:2 (2014), 63–76
3. benchmark Sinatra-like web frameworks — URL: https://github.com/mroth/phoenix-showdown
4. Results for js web frameworks benchmark – round 7 — URL: http://www.stefankrause.net/js-frameworks-benchmark7/table.html
5. Spectre.css — URL: https://picturepan2.github.io/spectre/index.html
6. Vue.js — URL: https://vuejs.org/
7. Basho Technologies. — URL: http://basho.com/products/\#riak
8. Phoenix — URL: http://phoenixframework.org/
9. Elixir — URL: https://elixir-lang.org/
10. Erlang Programming Language — URL: https://www.erlang.org/
11. Dygraph — URL: http://www.dygraphs.com
12. Kibana — URL: https://www.elastic.co/products/kibana
13. Datadog — URL: https://www.datadoghq.com/
14. Graphana — URL: https://grafana.com/

References

1. Zhivchikova N. S., Shevhcuk Y. V., Masshtabiruemaya raspredelennaya sistema arhivirovaniya sensornyh dannyh // Nauchniy servis v seti Internet: trudy XIX Vserossiyskoy nauchnoy konferentsii (18-23 sentyabrya 2017, Novorossiysk)

— Keldysh Institute of Applied Mathematics, 2017. — p. 157-169. — URL: http://keldysh.ru/abrau/2017/35.pdf doi:10.20948/abrau-2017-35
2. Stotskiy M. V., Shevchuk Y. V., Shablony v module vizualizatsii web-interfeysa sistemy monitoringa Botikmon // Programmnye sistemy: teoriya i prilozheniya, 5:2 (2014), 63–76
3. benchmark Sinatra-like web frameworks — URL: https://github.com/mroth/phoenix-showdown
4. Results for js web frameworks benchmark – round 7 — URL: http://www.stefankrause.net/js-frameworks-benchmark7/table.html
5. Spectre.css — URL: https://picturepan2.github.io/spectre/index.html
6. Vue.js — URL: https://vuejs.org/
7. Basho Technologies. — URL: http://basho.com/products/\#riak
8. Phoenix — URL: http://phoenixframework.org/
9. Elixir — URL: https://elixir-lang.org/
10. Erlang Programming Language — URL: https://www.erlang.org/
11. Dygraph — URL: http://www.dygraphs.com
12. Kibana — URL: https://www.elastic.co/products/kibana
13. Datadog — URL: https://www.datadoghq.com/
14. Graphana — URL: https://grafana.com/