ПРАКТИЧЕСКАЯ РАБОТА №11 – РЕАКЦИИ ПЛАТФОРМ ИНТЕРНЕТА ВЕЩЕЙ НА ПРИХОДЯЩИЕ ДАННЫЕ

Назначение реакций

Как уже было сказано ранее, платформы Интернета вещей используются для обработки данных и реализации логики работы IoT сервиса без привязки к конкретным физическим устройствам. Рассмотренный в предыдущей работе пример позволяет строить самую базовую логику по управлению устройствами, однако, никакой реакции от облачной платформы на ответы от физического устройства не предусмотрено, что не позволяет полноценно отслеживать правильность работы устройства.

IoT платформы имеют возможность отправлять оповещения (тревоги) при возникновении ошибок в ходе выполнения скриптов обработки приходящих данных. Данный механизм позволяет решить проблему отслеживания состояния физического устройства, которое передается в качестве ответа на отправляемый запрос о смене состояния. Помимо этого, можно использовать данный механизм для оповещений в случае поступления на устройство данных, выходящих за рамки допустимых значений.

Облачная платформа Rightech.io также поддерживает подобный механизм тревог-оповещений, следовательно, рассмотрим его на примере данной платформы.

Показать сообщение

Отправка сообщений позволяет следить за текущими изменениями, происходящими при исполнении логики. Если добавить данную команду на вход в какое-либо состояние, то при переходе объекта в данное состояние система отправит необходимое уведомление. Оно проинформирует о том, что совершилось определенное событие, вследствие которого объект теперь находится в данном состоянии. В итоге при регулярной отправке уведомлений вы всегда будете знать о ходе статусе процесса управления.

Формирование сообщений

Для формирования сообщения добавьте в состояние команду Показать сообщение. Справа появится боковая панель, на которой выберите степень важности сообщения:

- **Информационное;**
- **Важное;**
- **Критическое.**

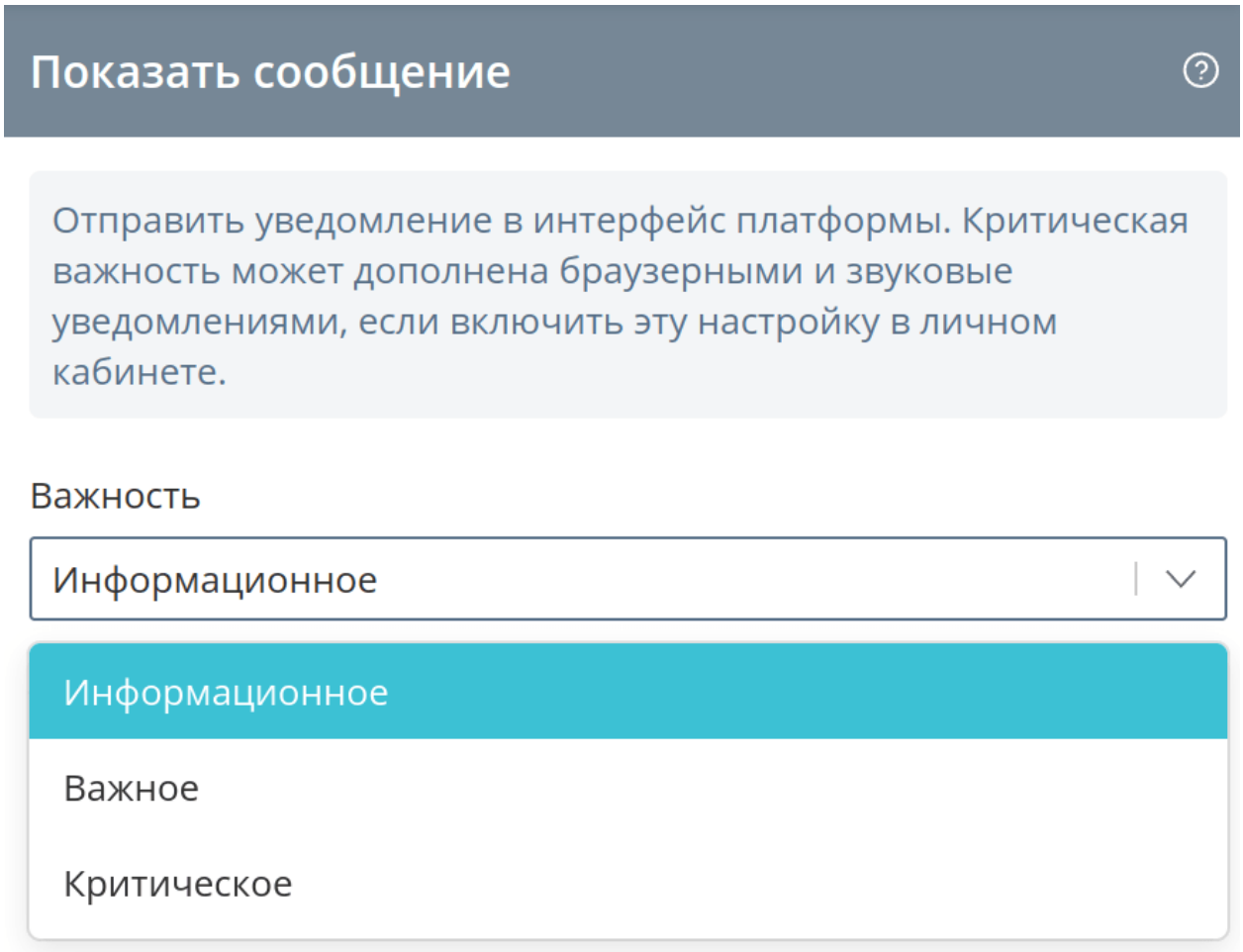

Рисунок 1. Типы сообщений

Если необходимо проинформировать о совершении какого-либо перехода или события, то тогда достаточно указать сообщение как информационное. Если же уведомление играет более значительную роль, например, зарегистрировано какое-то отклонение от нормы или нарушение, то имеет смысл повысить важность сообщения.

Далее заполните содержимое сообщения. Текст сообщения формируется в произвольном виде, его формат строго не регламентирован. Однако желательно, чтобы он был коротким и информативным для удобного отображения на платформе.

Помимо описательной информации о произошедшем событии, в сообщении можно отправить конкретные текущие значения параметров. Все значения, которые можно использовать, указаны в API link объекта.

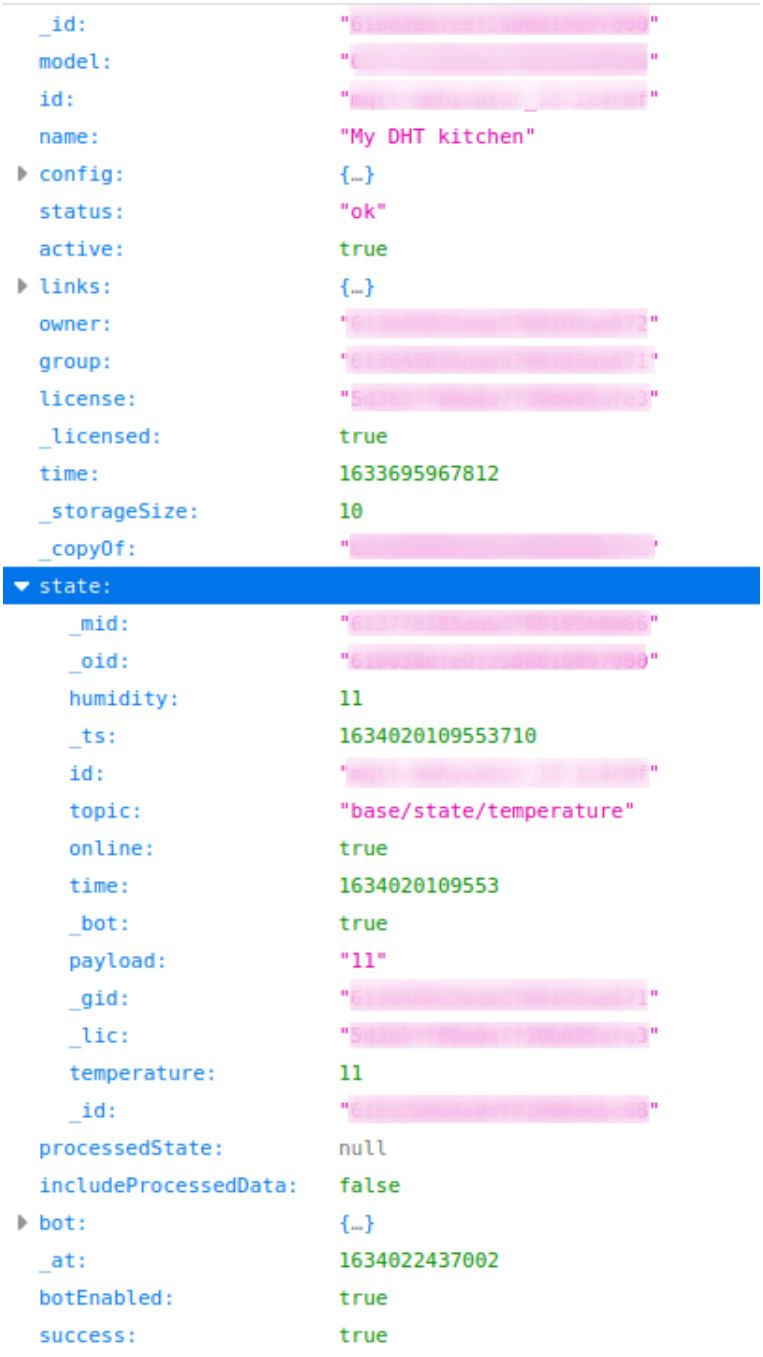

Рисунок 2. Api link

В этом случае в сообщении формируется конструкция {{object.<...>.parameter_id}}, где

<...> - подсистемы вложенности, в которых находится параметр, указываются через точку;

parameter_id - идентификатор параметра.

Например, {{object.state.humidity}}, {{object.name}} и т.д.

Тогда содержимое уведомления может быть сформировано следующим образом:

Рисунок 3. Текст сообщения

В тексте сообщения вместо {{object.state.temperature}} будет подставлено текущее значение параметра с идентификатором temperature.

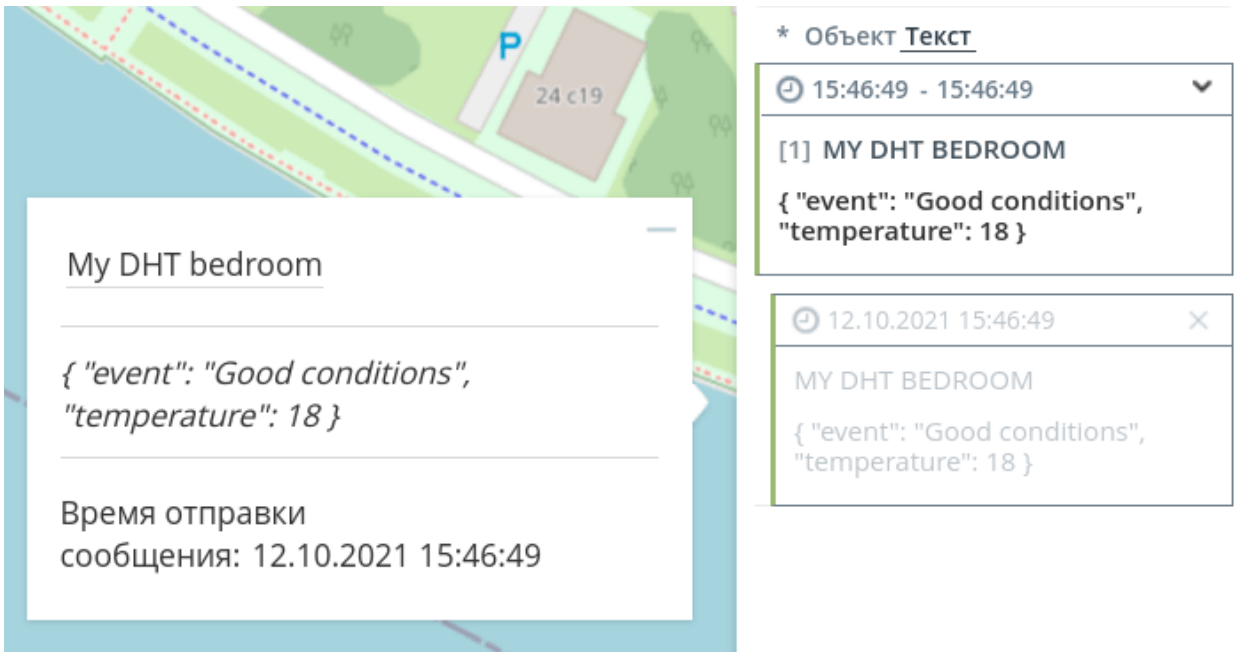

Рисунок 4. Вывод сообщения

Отображение и просмотр сообщений

Отправленные сообщения показываются в правой боковой панели сообщений.

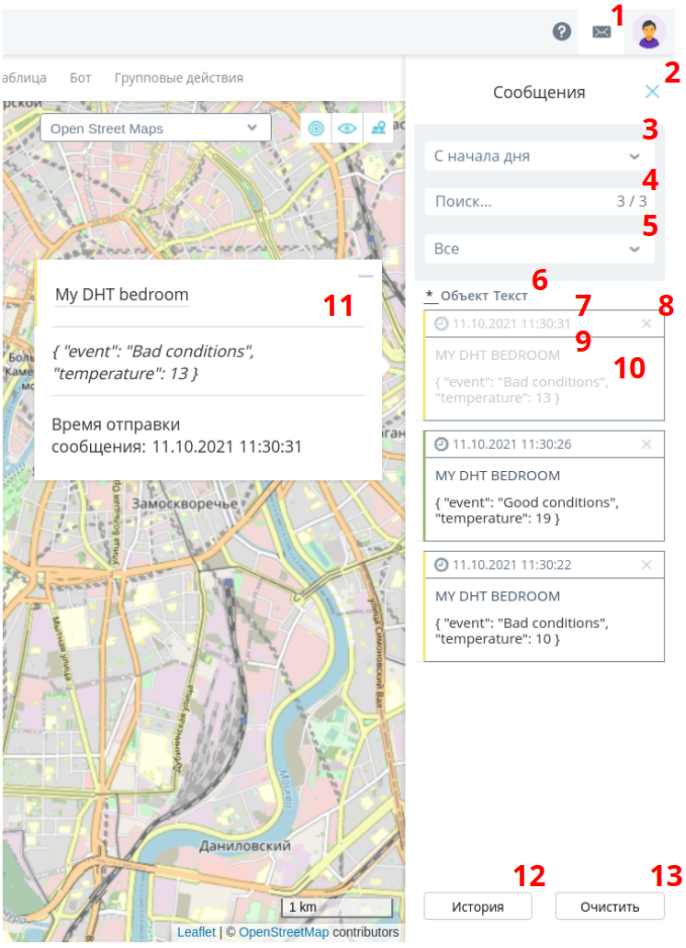

Рисунок 5. Отображение сообщения

1. Кнопка для открытия/закрытия панели сообщений. При наличии непрочитанных сообщений выглядит так

Рисунок 6. Кнопка панели сообщений

2. Кнопка для закрытия панели сообщений
3. Фильтр по времени получения: можно отобразить сообщения, которые были получены **С начала часа**, **С начала дня**, **С начала месяца** и **С начала года**
4. Строка поиска, поиск происходит как по объектам, так и по содержимому сообщения
5. Фильтр по степени важности: можно отобразить сообщения с уровнем **Критический**, **Важный** и **Информация**. При этом каждый уровень идентифицируется своим цветом: красным, желтым или зеленым соответственно
6. Группировка сообщений
7. Дата и время получения сообщения
8. Скрытие сообщения из списка
9. Имя объекта, к которому относится данное сообщение
10. Содержимое сообщения
11. Если кликнуть на отдельное сообщение, то оно полностью отобразится чуть левее боковой панели, будет помечено как прочитанное и исчезнет из списка сообщений
12. История сообщений
13. Очистка списка сообщений

Рассмотрим пример вывода сообщения при пересечении входящим параметром, заданного порога. Для этого необходимо установить дополнительный блок перехода, проверяющий соответствие параметров, в которые мы установим граничные значения для уровня температуры.

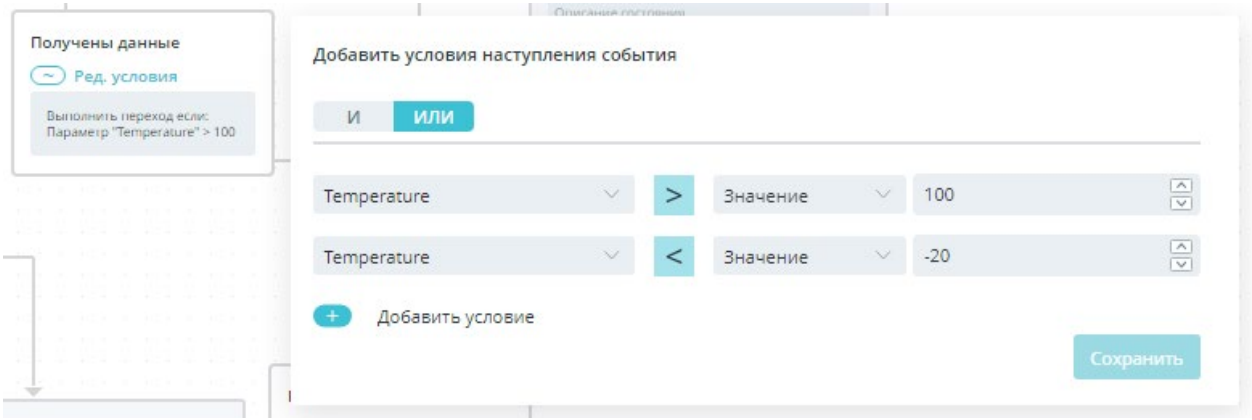

Рисунок 7. Предел значений

Так же для установки диапазонов можно использовать уровни, задаваемые в модели, в параметрах датчика температуры.

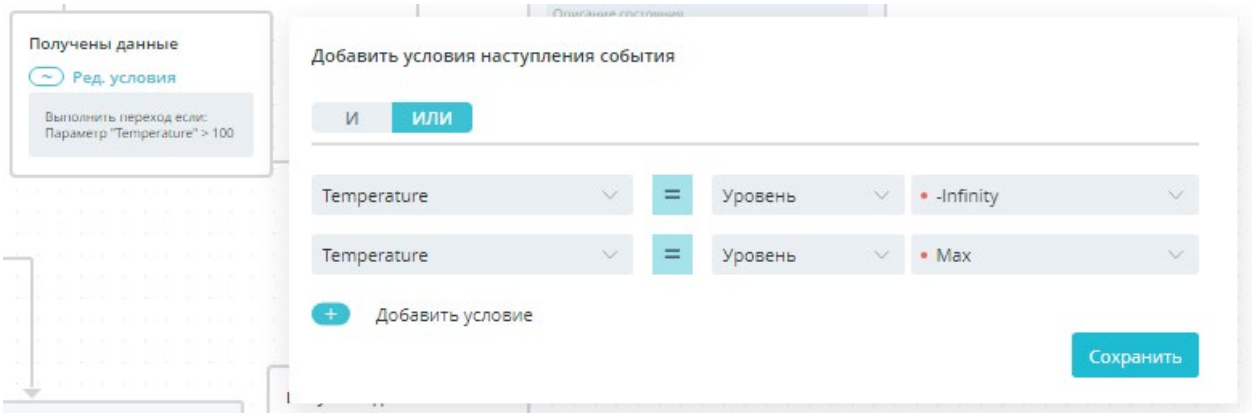

Рисунок 8. Предел уровней

При прохождении данного условия, задаем новое состояние системе, а именно генерацию события на входе «Показать сообщение», внутрь которого помещаем необходимый текст ошибки, перед этим обозначив её важность.

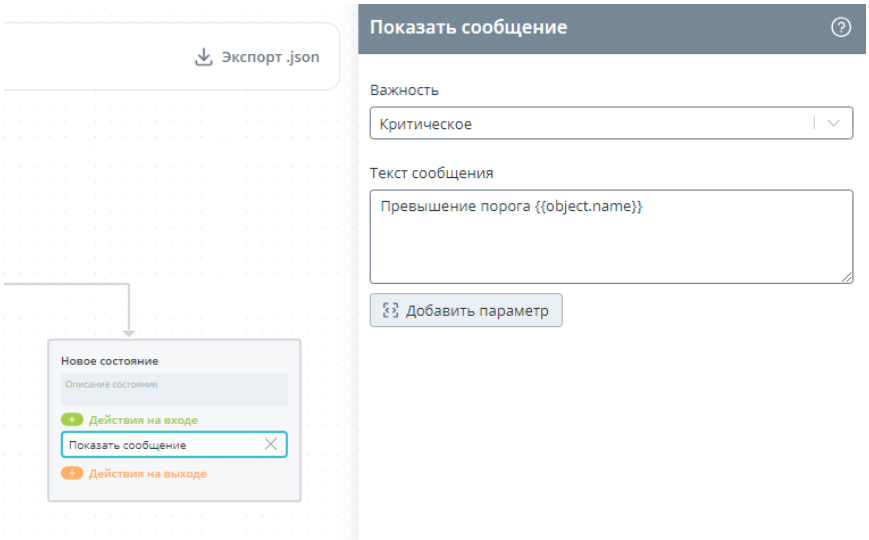

Рисунок 9. Создание сообщения

При передаче данных, выходящих за установленные пределы, получаем оповещение о критической ошибке.

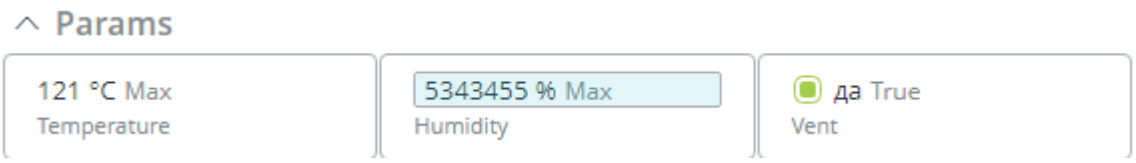

Рисунок 10. Полученные данные

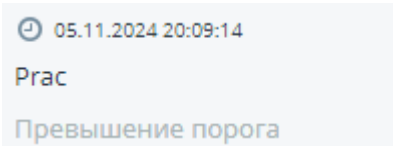

Рисунок 11. Сообщение

Для отслеживания типа пришедших данных воспользуемся функционалом обработчика. Для создания нового обработчика перейдите на вкладку **Обработчики**. Добавьте обработчик, нажав на плюсик.

Заполните следующие поля:

- **имя** — наименование обработчика;
- **описание** — подробная характеристика обработчика, заполняется при необходимости;
- **параметры по умолчанию** — поведение обработчика при получении пакета данных, в котором нет нужных входных аргументов.

Нажмите кнопку **Создать**. Перед вами откроется готовый шаблон с небольшим кодом для расчета суммы двух входных значений. Используйте его в качестве наглядного примера того, каким образом может быть реализован обработчик.

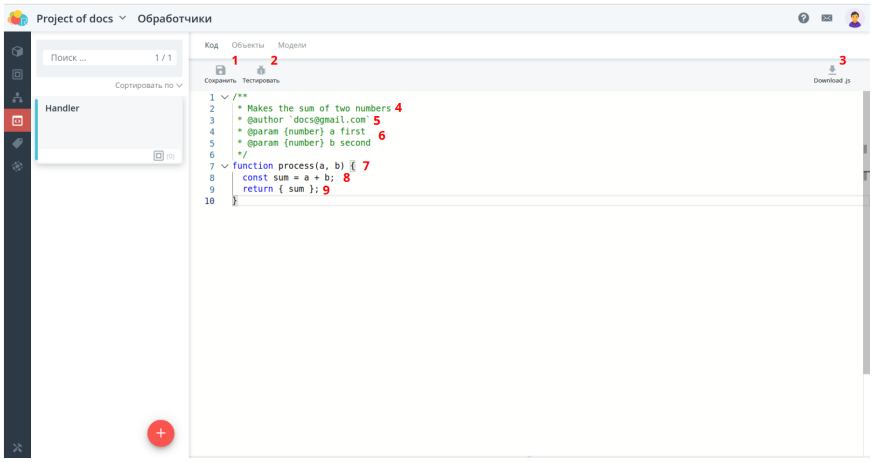

Рисунок 12. Шаблон обработчика

1. Сохранить: Сохранить изменения обработчика. Если ранее обработчик был запущен на объектах, он продолжит работать уже в обновленной версии
2. Тестировать: Открыть окно тестирования, в котором можно ввести входные параметры и проверить получившиеся выходные значения
3. Download .js: Экспортировать обработчик в файл формата JSON
4. В данной части комментария указывается дополнительное описание обработчика
5. В данной части комментария указывается логин автора обработчика
6. В данной части комментария указываются входные параметры в следующем формате: * @param {<тип_данных>} <имя> <описание>
7. В круглых скобках указываются имена входных параметров в следующем формате function process(*параметр_1*, *параметр_2*, ..., *параметр_n*)
8. Код обработчика, работающий с входными параметрами по заданному алгоритму
9. В фигурных скобках после слова return указываются имена выходных параметров в следующем формате return {параметр_1, параметр_2, ..., параметр_n};

Для проверки соответствия входящих данных, создадим в модели дополнительный параметр флаг chek. Далее перейдем в раздел обработчики и создадим новый обработчик.

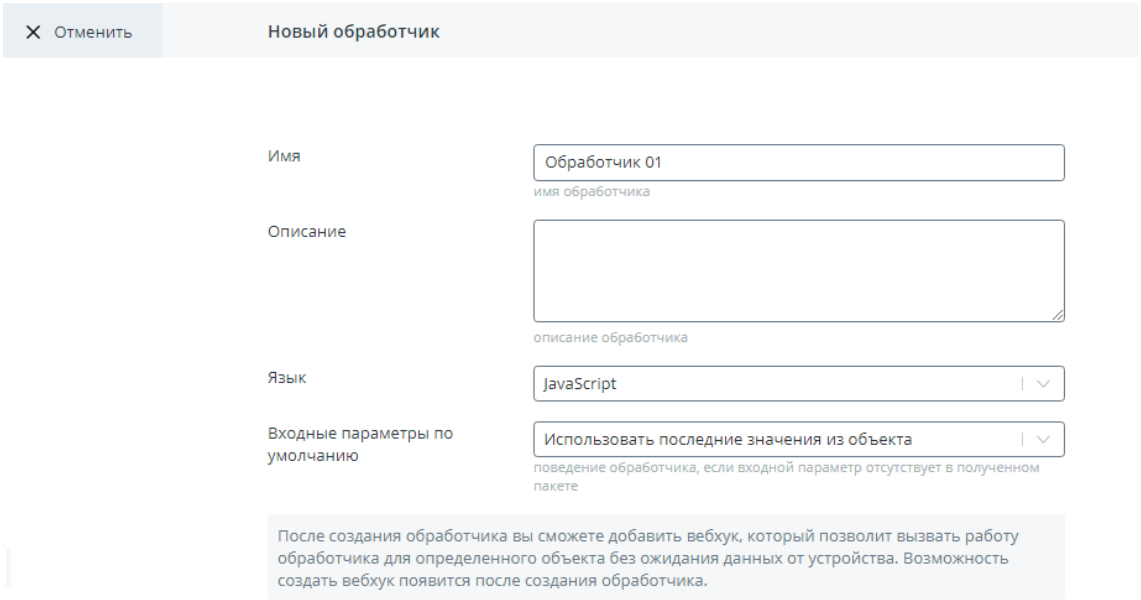

Рисунок 13. Окно создания обработчика

Присвоим нашему обработчику объект с данными, которого он будет взаимодействовать.

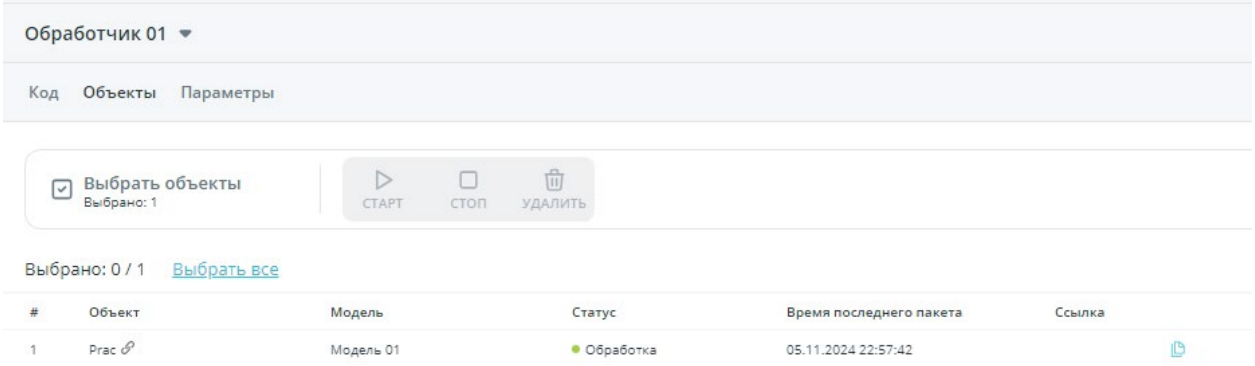

Рисунок 14. Выбор объекта

В разделе параметры добавим ссылки на входные и выходные параметры

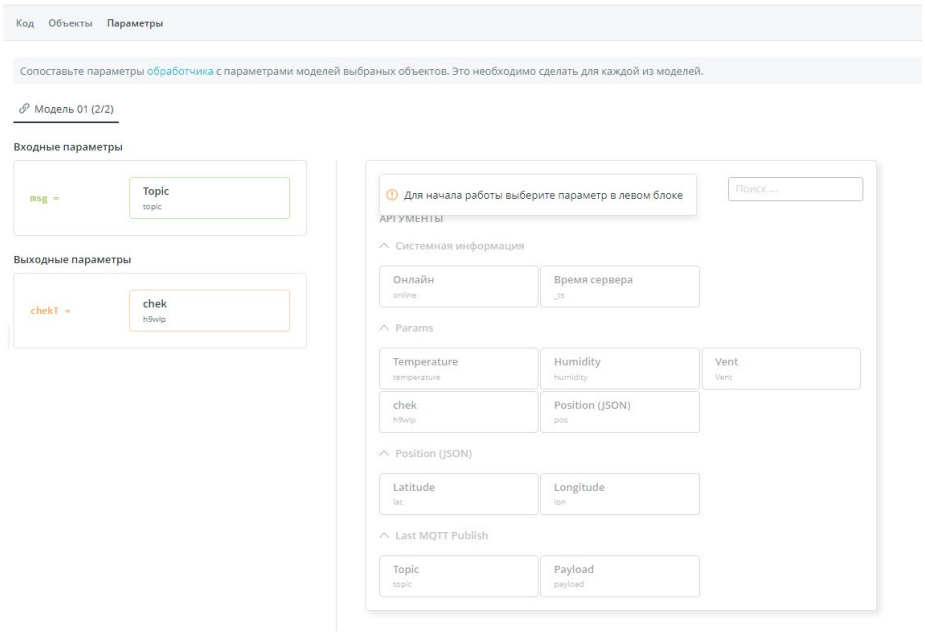

Рисунок 15. Параметры

Далее создадим функцию, и передадим в неё последние пришедшие нам данные. С помощью условия проверим, соответствие этих параметров с необходимыми нам данными.

```
function process(msg) {

    let chekT;
    if (msg !== "base/state/temperature") {
        console.log("Параметр msg не удовлетворяет условию");
        chekT = true;
    } else {
        console.log("Параметр msg соответствует топику 'base/state/temperature'");
        chekT = false;
    }
    return { chekT };
}
```

Рисунок 16. Код обработчика

Так же возможно генерировать собственные события для отслеживания получаемых данных с помощью функции *ric.events.gen("Название события", { value });*

Созданные события отобразятся в разделе «Объекты» во вкладке «События».

Следующим шагом для генерации сообщения зададим условие проверки входящих данных, по аналогии с проверкой допустимого диапозона.

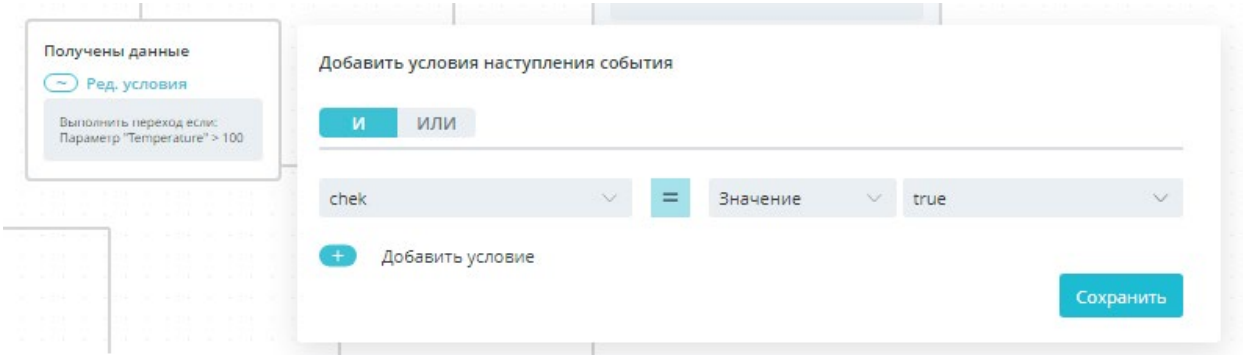

Рисунок 17. Проверка условия

Задание практической работы №11

Добавьте в логику правил из 10 практической работы формирование нескольких типов тревог:

- Для первого объекта логики из варианта – тревогу при выходе приходящего параметра за допустимые границы (границы задайте самостоятельно и проверьте работоспособность на физическом датчике);
- Для второго объекта логики из варианта – тревогу при отсутствии ожидаемого параметра в приходящем сообщении (к примеру, в приходящем сообщении отсутствует параметр с состоянием кнопки);

В отчет включите обновленные цепочки правил, скрипты проверок и формирования тревог, а также результаты тестирования цепочек при помощи утилит mosquitto.