УДК 004.89:728

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ УПРАВЛЕНИЯ ЭНЕРГЕТИЧЕСКИМИ РЕСУРСАМИ ЖИЛОГО ПОМЕЩЕНИЯ

Маргун Д.К.
Научный руководитель – Юденков В.С., к.т.н., доцент

Цель работы состоит в разработке программного обеспечения системы «умный дом» [1]. Основными задачами являются: 1) анализ существующих систем; 2) разработка протоколов коммуникации устройств внутри системы; 3) разработка средств визуализации информации, полученной с датчиков.

В качестве канала связи между сервером и датчиками используется протокол TCP/IP [2]. Данные системы передаются на сервер в формате бинарного потока на самом низком уровне для обеспечения максимальной производительности. Структура циркуляции информации внутри системы может быть описана следующим образом: Датчик(и) → Сервер → Браузер. Полученные данные от датчиков кэшируются на сервере. Обновление данных происходит при изменении показаний датчика. Такая модель обмена данными позволяет значительно снизить трафик внутри сети. Логика сервера, обрабатывающего поступающую информацию, создана в виде аддона для асинхронной платформы Node.js. В качестве базы данных была выбрана нереляционная технология MongoDB. Визуализация данных выполняется с помощью новейших Web-технологий: React, Spring, React-admin. Вид главной страницы представлен на рисунке 1.

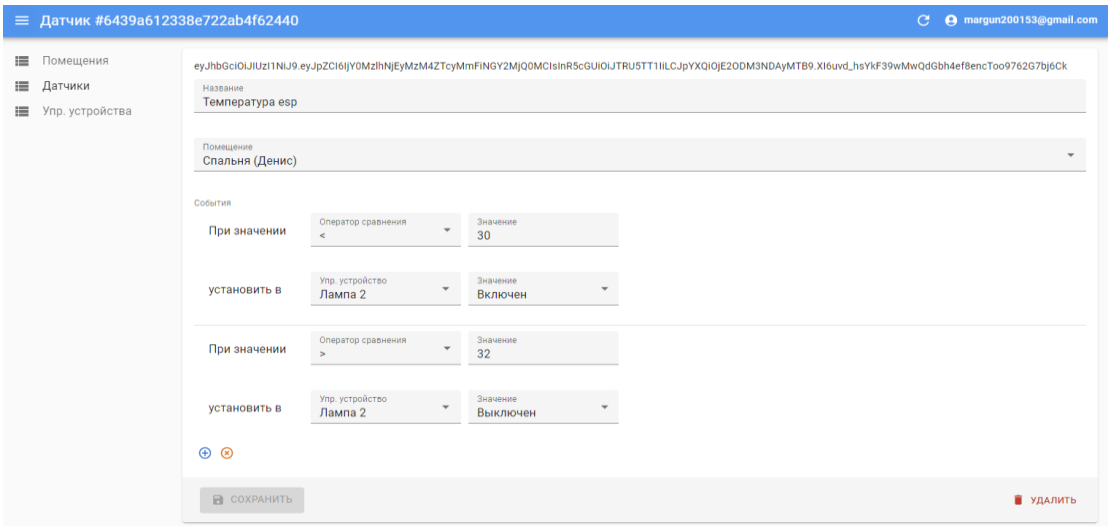

Рис. 1. Вид главной страницы.

Для быстрой загрузки сайта все страницы системы состоят из базовых компонентов. Построение окончательного вида страницы осуществляется на стороне клиента. Такой подход позволяет значительно уменьшить размер веб-сайта, как итог: моментальная загрузка сайта.

Реализованная система состоит из клиентской части, сервера и провайдеров информации (датчиков): датчик температуры, датчик давления, датчик закрытия-открытия, и т.д. Система получает информацию с датчиков в реальном времени (Рис.2).

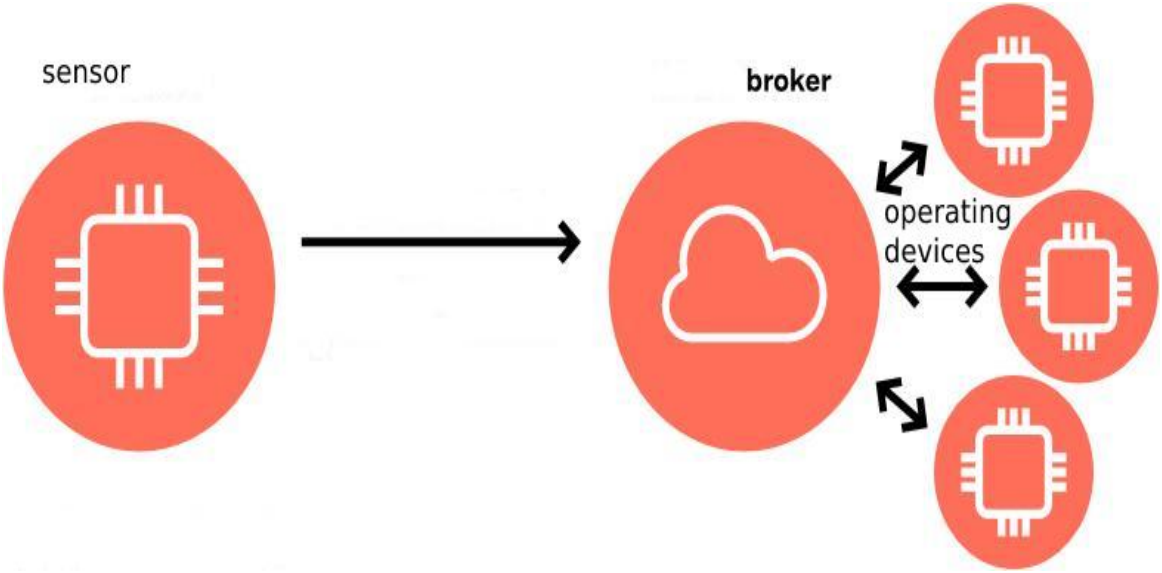

Рис.2. Информационные потоки

В качестве языка программирования для сервера используется Java и фреймворк Spring. В качестве системы сборки выбран maven. Maven использует файл pom.xml в котором указаны зависимости и процесс сборки программы. Maven использует эту информацию для автоматического загрузки и подключения нужных зависимостей из центрального репозитория или других источников. Таким образом, вам не нужно вручную скачивать и подключать каждую зависимость — Maven автоматически управляет этим процессом.

На сервере используются следующие бибилотеки:

– spring-boot-starter-data-mongodb - предоставляет зависимости для подключения и использования MongoDB в проекте на Spring Boot.
– spring-boot-starter-actuator - предоставляет зависимости для мониторинга и управления приложением, включая информацию о состоянии, метрики и т.д.
– spring-boot-starter-web - предоставляет зависимости для разработки веб-приложения с использованием Spring MVC, включая поддержку HTTP-запросов, контроллеры и т.д.

– spring-boot-devtools - предоставляет инструменты разработчика для автоматической перезагрузки при изменениях кода в процессе разработки. Зависимость имеет область runtime, что означает, что она используется только при запуске приложения в режиме разработки.
– lombok - предоставляет аннотации и инструменты для упрощения разработки Java-кода, такие как автоматическая генерация геттеров, сеттеров, конструкторов и т.д.
– spring-boot-starter-data-jpa - предоставляет зависимости для работы с JPA (Java Persistence API) и базами данных через ORM (Object-Relational Mapping).
– spring-data-commons - предоставляет общие зависимости для проектов, использующих Spring Data, такие как поддержка репозиториев и механизмы работы с данными.
– querydsl-core, querydsl-apt, querydsl-jpa - предоставляют зависимости для работы с Querydsl, который предоставляет возможности типобезопасного запроса и манипулирования данными через JPA.
– spring-boot-starter-security - предоставляет зависимости для реализации безопасности в приложении с использованием Spring Security.
– spring-boot-starter-validation - предоставляет зависимости для валидации данных и проверки их соответствия определенным правилам.
– Jjwt - предоставляет зависимость для работы с JSON Web Tokens (JWT), используемыми для аутентификации и авторизации в веб-приложениях.
– mapstruct: Предоставляет зависимость для генерации автоматического маппинга объектов между различными слоями приложения.
– spring-websocket: Предоставляет зависимость для поддержки веб-сокетов в приложении на базе Spring. Веб-сокеты позволяют установить постоянное двустороннее соединение между клиентом и сервером, что позволяет обмениваться данными в режиме реального времени.
– spring-messaging: Предоставляет зависимость для поддержки механизмов обмена сообщениями в приложении на базе Spring. Эта зависимость включает в себя функциональность для отправки, получения и обработки сообщений между различными компонентами приложения, в том числе через веб-сокеты.

Были выделены следующие POJO модели:

1. FlatEntity – класс являющейся отоброжением помещений;
2. OperatingDeviceEntity - класс являющейся отоброжением действующих устройств;
3. SensorEntity - класс являющейся отоброжением сенсоров;
4. UserEntity – клас являющейся отоброжением пользователей.

Для них были созданы репозитории:

1. FlatRepository;
2. OperatingDeviceRepository;

3. SensorRepository;
4. UserRepository.

А также классы сервисы:

1. FlatService;
2. OperatingDeviceService;
3. SensorService;
4. UserService;
5. AuthService – сервис необходимый для авторизации.

Для доступа из сети были разработаны контроллеры:

1. FlatController;
2. OperatingDeviceController;
3. SensorController;
4. AuthController.

Литература

1. Умный дом. Википедия. [В Интернете] [Цитировано: 10 4 2017 г.] https://ru.wikipedia.org/wiki/Умный_дом.
2. Тим Паркер, Каранжит Сиян. 2004. TCP/IP. Для проффесионалов. 3-е. Минск: Питер, 2004. 5-8046-0041-9.

УДК 621.350.11

СИМУЛЯТОР ПОДВИЖНОГО ОБЪЕКТА НА БАЗЕ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА

Вдовухин М.А.
Научный руководитель – Юденков В.С., к.т.н., доцент

Гибридными электрическими транспортными средствами принято считать механизмы, приводимые в движение с использованием механической энергии различных источников, как правило это электродвигатель и двигатель внутреннего сгорания.

Рассмотрим гибридные транспортные средства на примере автомобилей. В начале зарождения автомобильной индустрии автомобили строились с электроприводом. Это объяснялось лучшими эксплуатационными характеристиками по сравнению с паровыми машинами, а также двигателями внутреннего сгорания (ДВС). А именно необходимость присутствия трансмиссии приводила к значительному увеличению физических размеров и массы конструкций. Электродвигатели же были значительно меньше по габаритам и весу. Различались и источники питания. В начале 20-го века технологии накапливания электрической