УДК 004.415.2 10.5281/zenodo.3904148

Д. Авельцов, В. Гайдамако, *dolpha@gmail.com*
Институт машиноведения и автоматики НАН КР, Бишкек

РАЗРАБОТКА МОДУЛЯ ВИРТУАЛИЗАЦИИ СЕНСОРНЫХ УСТРОЙСТВ ДЛЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ СИСТЕМ

В настоящей работе рассмотрен пример разработки модуля виртуализации сенсорных устройств для использования в Интернете вещей, распределенных и облачных информационно-измерительных системах, предоставляющих измерительные устройства в качестве услуги. При реализации системы использовались стандарты описания датчиков OGC. Также на основе данных стандартов имплементированы модели данных и связи между ними. Разработан модуль взаимодействия с пользователем и проведено тестирование.

Ключевые слова: сенсорные устройства, датчики, виртуализация, микросервисы. виртуализация датчиков, docker, docker-контейнер, микросервисы (microservices), OGC (Open Geospatial Consortium), sensorML, Интернет вещей, информационно-измерительные системы (ИИС)

Введение

В современном мире, мире Интернета Вещей (ИВ), умных домов и умных городов, умные устройства – сенсоры и актуаторы, используются широко и повсеместно, и стремительный рост их числа приводит к росту потребности в решениях, позволяющих подключать эти устройства к сети, собирать, хранить и анализировать их данные. Применение облачных технологий позволяет различным пользователям и приложениям получать совместный (разделяемый) доступ к устройствам и к их данным через технологию виртуализации [1-6]. Для физического устройства создается множество его виртуальных представлений, свое для каждого конкретного пользователя. Виртуальное устройство может отображать не одно, а комбинации нескольких физических устройств в зависимости от запросов пользователей. Использование виртуальных устройств вносит в систему дополнительный слой абстракции, что позволяет масштабировать систему и работать с устройствами, не заботясь об особенностях их физической реализации. Виртуализация датчиков используется в программно-аппаратных системах, реализующих подходы Sensor-Cloud Infrastructure (SCI), Information-as-a-Service (IaaS), Framework of Sensor-Cloud Integration (FSCI), Virtual Federated Sensor Network (VFSN), Sense-Cloud [5].

При организации совместного доступа важна стандартизация – стандарты описания и управления устройствами, стандарты для форматов и структуры данных, генерируемых этими устройствами, становятся острой необходимостью. Было создано множество стандартов для описания устройств. Одной из организаций, занимающихся стандартизацией, является OGC – Open Geospatial Consortium (Открытый Консорциум Геопрстранственной информации), международная некоммерческая организация, созданная в 1994 году для разработки стандартов геопространственных данных, учета и согласования координатных систем, используемых в мире. В настоящее время координирует деятельность более 500 [7] правительственных, коммерческих, некоммерческих и научно-исследовательских организаций с целью разработки и внедрения консенсусных решений в области открытых стандартов не только для геопространственных данных, обработки данных геоинформационных систем, но и

других стандартов в различных областях, особенно там, где важны данные о положении в пространстве. Стандартами OGC, связанными с описанием и использованием сенсорных устройств, являются Semantic Sensor Network Ontology (Стандарт описания сенсорных сетей), SensorML 2.1 – язык моделирования сенсорных устройств. SensorML предоставляет модели и XML-кодировки для описания сенсоров, актуаторов и процессов, позволяет реализовать описание спецификации обслуживания устройства, описание входных данных, выполнение агрегатных процессов и рабочих процессов по требованию и потоковую передачу данных. Стандарты ISO [8] используют OGC стандарт gml (Geography Markup Language) для описания устройств и данных.

В данной статье рассматриваются вопросы разработки программных средств виртуализации сенсорных устройств для создания виртуальной сенсорной среды и изоляции приложений-клиентов от оборудования.

Инфраструктура распределенной/облачной информационно-измерительной системы

Стандартная модель инфраструктуры распределенной/облачной Информационно-Измерительной Системы (ИИС), которая используется и в Интернете Вещей (ИВ, англ. Internet of Things, IoT), представлена на рисунке 1 [1, 5]. Данные передаются от сенсорных устройств, датчиков различных типов, объединенных в Беспроводные Сенсорные Сети (БСС) к концентраторам (в литературе также может иметь названия: шлюз, хаб – устройство для связи с глобальной сетью, может исполнять различные дополнительные функции, в том числе выступать в качестве частного облака), далее информация передается на облачную платформу, которая также может быть интегрирована с другими сервисами/платформами. Соответствующие модули обеспечивают связь с пользовательскими приложениями на различных типах устройств – стационарных десктопах, мобильных устройствах, специализированных компьютерах.

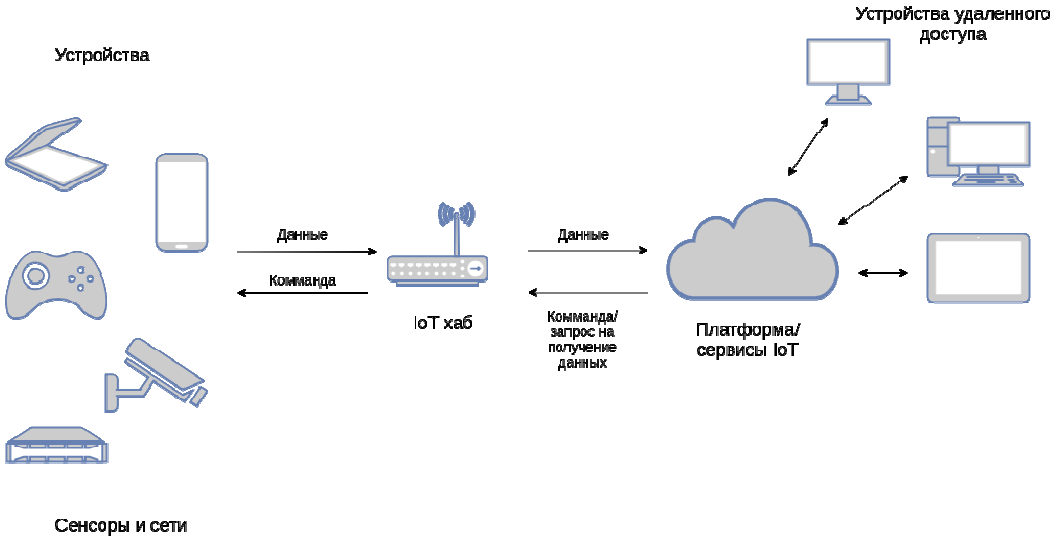

Рисунок 1 – Инфраструктура Интернета вещей

Упрощенная модель архитектуры сети распределенной/облачной ИИС представлена на рис. 2. Типы сетей для каждого уровня зависят от реализации устройств – участников данной сети, а также от иных факторов, влияющих на возможность использовать те или иные типы сетей.

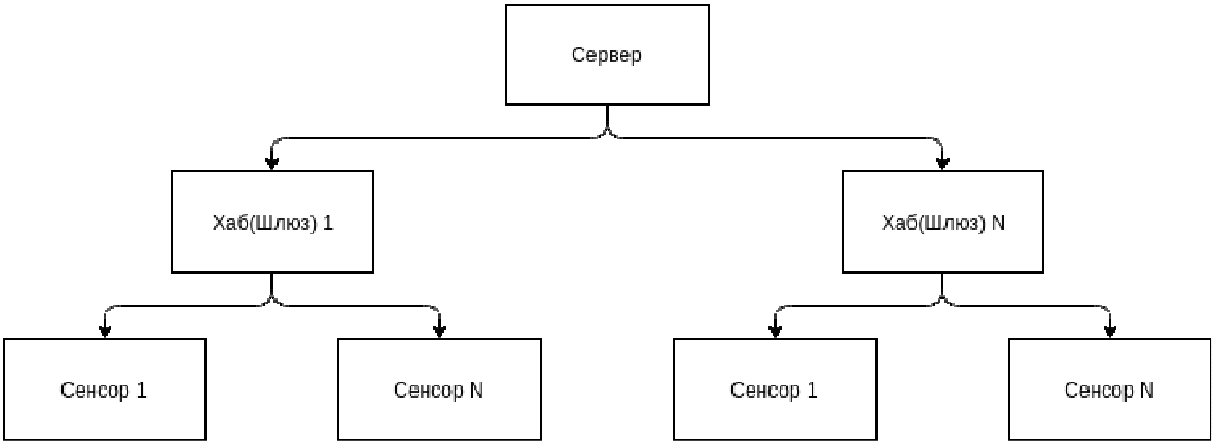

Рисунок 2 – Схема архитектуры сети распределенной ИИС

Общая архитектура решения для ИВ или (если исключить актуаторы) для распределенной/облачной ИИС представлена на рис. 3. Физические сенсорные устройства (датчики) регистрируют параметры окружающей среды, формируют аналоговый сигнал, который АЦП (аналого-цифровой преобразователь) преобразует в цифровой. После конвертации информация обрабатывается локальным процессором периферийного устройства, в частности, ставится метка-идентификатор (тег), обязательно включающий временную метку(timestamp).

Следующее звено – коммутатор (хаб) может находиться как в облаке, так и на периферии. Коммутатор перенаправляет полученную информацию на различные объекты, используя теги. В роли объектов могут выступать различные сервисы, очереди или хранилище. Так происходит получение и обработка информации от конкретного периферийного устройства. Далее на уровне интегратора полученная информация от различных периферийных устройств суммируется по однотипным тегам, при этом типы устройств могут быть различными. Информация от всех объектов систематизируется аналитическим блоком, который может включать Искусственный Интеллект (ИИ), машинное обучение и прочее. Блок презентации отвечает за взаимодействие с пользователем системы.

Любая распределенная система требует механизмов гарантированной доставки информации. В блок виртуального представления периферийного устройства – ту часть, которую представляет собой разрабатываемая система, записывается информация, принятая (или передаваемая в случае актуатора) от периферийного устройства. В случае неудачной попытки передачи информации от периферийного устройства в облако, либо от облака к периферийному устройству, осуществляются повторные попытки передачи и кэширование данных на уровне виртуального устройства.

Требования к системе

Целью представляемой работы было создание прототипа модуля виртуализации датчиков для распределенной/облачной информационно-измерительной системы и ИВ и безотказной передачи данных в сетях различной топологии.

Разрабатываемая система может применяться для виртуализации датчиков, для получения информации, собранной датчиками, для получения информации о датчиках, а также поиска информации и датчиков.

Модуль виртуализации датчиков должен быть независимой системой, предоставляющей пользовательский интерфейс для описания и управления датчиками, а также API (Application Programming Interface, программный интерфейс приложения) для любых авторизованных клиентских приложений.

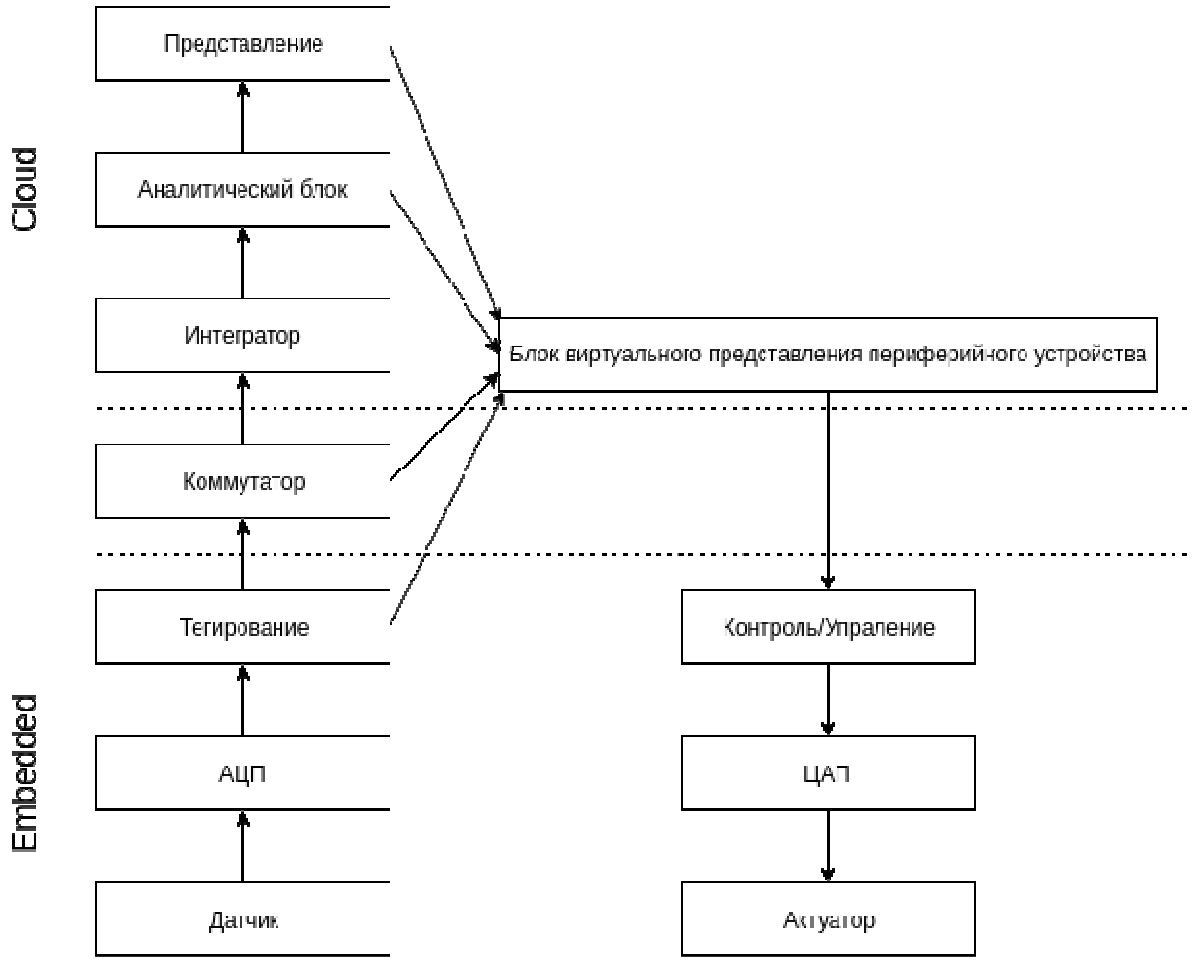

Рисунок 3 – Общая архитектура решения для распределенной системы. Embedded – реализация на стороне устройства, Cloud – реализация в облачной инфраструктуре

Система должна выполнять следующие функции:

- Регистрация / авторизация пользователя;
- Создание, редактирование, экспорт шаблонов на языке SensorML при помощи веб-интерфейса;
- Виртуализация, создание виртуального представления датчиков на основе шаблона SensorML;
- Обработка запросов от датчиков;
- Получение данных с физических датчиков через слой виртуализации;
- Хранение информации, полученной от датчиков; хранение информации должно происходить как на уровне периферийного устройства, так и в облаке. Периферийное устройство сохраняет свои настройки, состояние и временно хранит информацию (кэширует), пока данные гарантированно не переданы в облако
- Кэширование информации, полученной от датчиков;

Система разрабатывается для 3 групп пользователей:

- Администратор системы. Имеет общее представление об аспектах использования системы;
- Владелец датчика. Должен знать технические характеристики своего датчика, быть способным описать его;
- Пользователь (клиент) системы. Особые требования не предъявляются;

Система должна предоставлять следующие интерфейсы:

- Пользовательский интерфейс для создания шаблонов датчиков и управления датчиками для владельца устройства;
- Программный интерфейс для получения информации о датчиках и информации, собранной датчиками;

Функциональные требования к системе отражены на диаграмме, представленной на рисунке 4 (Диаграмма вариантов использования).

Внешние требования к интерфейсам

Взаимодействие с пользовательской частью системы осуществляется с помощью HTTP-запросов.

Взаимодействие датчиков с системой осуществляется при помощи протоколов HTTP/S, TCP, HTTP 2.

На серверной машине установлена операционная система семейства linux, компилятор языка go версии 1.14 или выше, Docker Engine версии 19.03 или выше; на клиентской и серверных машинах есть доступ к сети Internet;

Безопасность. Схема распределения ключей шифрования – главная особенность, отличающая сети ИВ от обычной mesh-сети. Все устройства являются крайне недоверенными – любое устройство может быть скомпрометировано, поэтому ни одно устройство не может хранить мастер-ключ всей системы.

- Включение нового устройства в сеть должно происходить децентрализовано (на случай отсутствия связи с облаком).
- Необходимо использование протоколов, использующих схемы разделения секрета и слепой подписи.
- Система должна хранить логи и историю работы всех компонентов программной системы за последний месяц;
- Любые сбои в программной системе должны протоколироваться;
- Система должна быть защищена от несанкционированного доступа к данным и коду;
- Каждый датчик должен иметь свой уникальный идентификатор, который позволит идентифицировать его в системе;
- Каждый датчик имеет свою уникальную конфигурацию, которая описана в шаблоне, общее представление о подлинности датчика должно складываться из его идентификатора и данных его конфигурации;

Стандартизация. Система должна обслуживать различные устройства, компоненты и процессы. Для стандартизации описания требуется выбрать утвержденный стандарт описания устройств. SensorML (Sensor Model Language) является стандартом, одобренным Открытым консорциумом геопространственных данных – Open Geospacial Consortium (OGC) [7]. SensorML обеспечивает стандартные модели и в XML-кодирование [9] для описания датчиков и измерительных процессов. SensorML может быть использован для описания широкого спектра датчиков, в том числе динамических и стационарных платформ, и как in-situ, так и дистанционных датчиков.

Подходы к проектированию системы

В разрабатываемой системе используется как монолитный, так и микросервисный подход в проектировании. Пользовательская часть написана с применением монолитного подхода. Основное ядро системы спроектировано с применением

микросервисного подхода [10]. Такое решение продиктовано необходимостью масштабируемости системы. также через микросервисы можно объединять разные технологии, выбирая лучшие из возможных решений. Так как микросервисы являются независимыми друг от друга, стабильность системы повышается. Сбои и дефекты в одном микросервисе не повлияют на работу остальных, поэтому сама система будет функционировать с минимальными простоями. Всю систему условно можно поделить на модули (рис. 5): веб-сервер (Web-Server) и пользовательский интерфейс, платформа виртуализации (микросервисы) и виртуальные сенсорные узлы (Virtual Sensor Nodes).

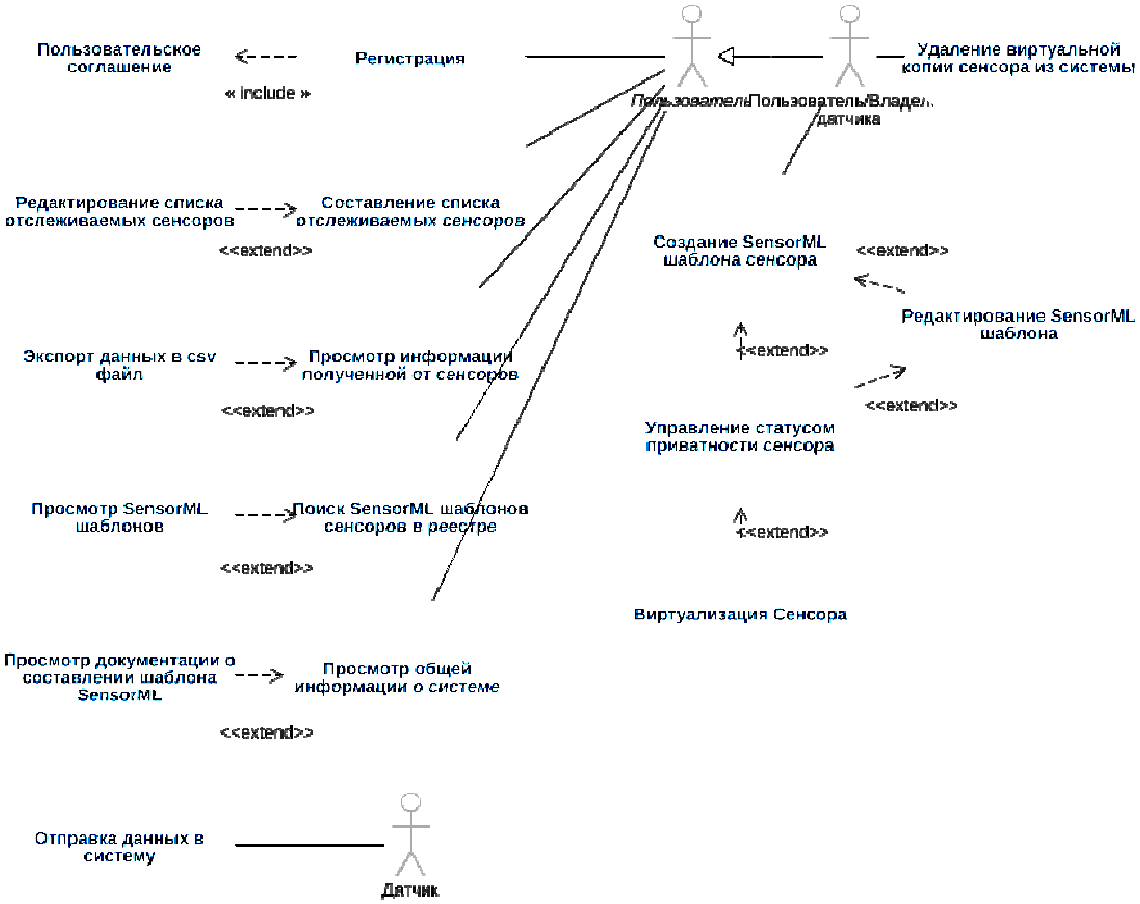

Рисунок 4 – Функции системы виртуализации сенсорных устройств
(Варианты использования)

Пользовательский интерфейс и веб сервер (Web Server)

Пользовательский интерфейс написан на языке typescript с использованием фреймворка Angular 7. Компоненты пользовательской части:

- api. Отвечает за взаимодействие с серверной частью. Взаимодействие происходят в соответствии со стандартом REST;
- editor. модуль SensorML редактора;
- model. модели стандарта SensorML;
- create. модуль загрузки и создания SensorML шаблона;
- services. Общие сервисы приложения;
- user. Пользовательский модуль, отвечает за регистрацию и авторизацию;

Для взаимодействия с серверной частью было решено использовать REST (REpresentational State Transfer) [11] архитектуру. Архитектура REST разработана для соответствия протоколу HTTP [12], используемому в сети Интернет. Вызов удаленной процедуры представляет собой обычный HTTP-запрос («REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

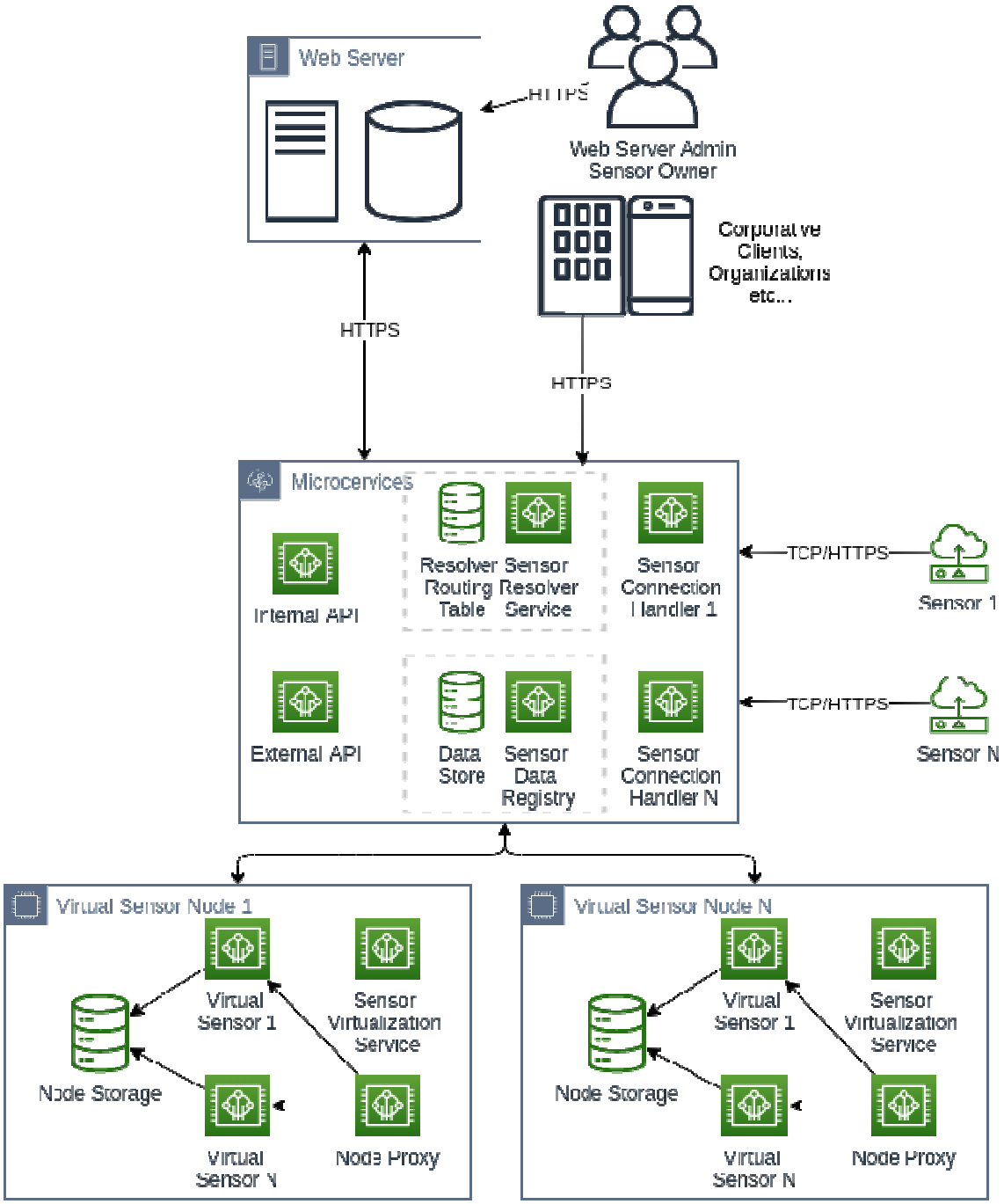

Рисунок 5 – Общая архитектура системы виртуализации сенсорных устройств

Архитектура REST очень проста в плане использования. Данные, передаваемые в теле запроса, могут быть в XML [9], JSON [13] формате, или же они могут быть переданы с помощью аргументов в URL (Universal Resourse Locator). В

разрабатываемой системе решено использовать JSON формат для изменения состояния системы и аргументы URL для методов получения данных.

Серверная часть написана на языке PHP 7.4 с использованием фреймворка Symfony 4.4. Данный компонент системы обслуживает пользовательский интерфейс и перенаправляет запросы виртуализации в платформу.

Архитектура Web Server содержит три слоя:

- слой доступа к базе данных, основанный на объектно-ориентированном отображении (ORM, Object Relational Mapping), содержит классы-сущности, отображающие состояние базы данных в состояние объектов этих классов-сущностей;
- слой бизнес-логики, содержащий всю логику обработки и изменения сущностей, проецируемых в базу данных;
- слой контроллеров, представляющий собой контроллеры для обработки запросов; представления определяются форматом данных, используемым для обращения к контроллерам.

Основные классы-сущности – это классы User и Sensor. Класс User представляет информацию о зарегистрированном пользователе, класс Sensor – информацию о созданном датчике, его шаблон, статус приватности, статус виртуализации.

Слой бизнес-логики описывает алгоритмы создания, получения, изменения, удаления для каждой сущности, а также алгоритмы обработки данных, представленных объектом сущности. Эти алгоритмы содержатся в классе, соответствующем классу-сущности, который они обрабатывают. В качестве параметров и возвращаемых результатов классы бизнес-логики используют экземпляры классов-сущностей.

Слой контроллеров представляет собой граничные классы, предоставляющие API интерфейс. Каждый граничный класс содержит методы обработки запроса. Запросы соответствуют сценариям использования системы. Основные граничные классы системы представлены на рис. 6. Методы этих классов должны извлекать параметры запроса, использовать методы бизнес-логики для обработки данных и возвращать результаты обработки в соответствующем JSON-формате. Никакой логики обработки данных контроллеры содержать не могут.

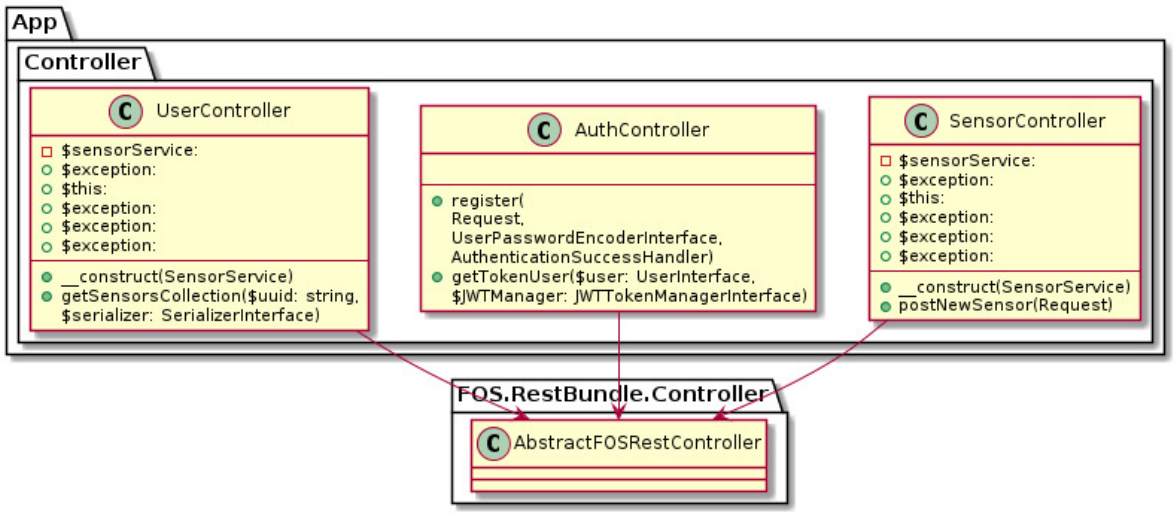

Рисунок 6 – Слой бизнес логики. Контроллеры (Диаграмма классов)

В качестве параметров и возвращаемых результатов граничные классы используют данные в формате JSON или полученные аргументы URL-запроса.

Диаграмма развертывания пользовательского модуля интерфейса и веб сервера (Web Server) описывает физические узлы в наиболее типичной конфигурации для поставщика услуг “виртуализация датчиков для распределенных измерительных систем” (рис. 6):

- Load Balancer Server – Балансировщик нагрузки распределяет трафик между серверными узлами веб-приложения, когда запущено более одного экземпляра сервера. К балансировщику нагрузки нет никаких особых требований. В качестве примера используется обратный прокси-сервер nginx.
- Web Application Server – Сервер веб приложения, обеспечивает пользовательский доступ к общей информации о системе, созданию шаблонов сенсоров при помощи графического интерфейса, поиску шаблонов сенсоров, просмотру информации о сенсорах, получаемой измерительной информации от сенсоров. Также обеспечивает администратору системы доступ через графический интерфейс для редактирования общей информации о системе и для администрирования пользователей, шаблонов и виртуальных сенсоров.
- Web application RDBMS Server – Сервер с установленным сервером баз данных. Используется для хранения данных пользователей и общей информации о системе для отображения в веб приложении.
- Sensor Template Registry Module / Sensor Virtualization Module – модуль реестра шаблонов датчиков / модуль виртуализации, не вошли в диаграмму на рис.7.

Пользовательский интерфейс используется для взаимодействия пользователя (владельца сенсора, администратора) с системой. Создание, редактирование и удаление шаблона сенсорного устройства на языке SensorML реализовано при помощи редактора шаблонов. В текущей версии системы пользователь может также объединить компоненты в виде датчиков в физические системы – например, описав термометр, анемометр и процесс вычисления фактора ветра (обычно при помощи него вычисляют температуру “как ощущается”), можно объединить все эти компоненты в погодную станцию и не собирать данные каждого сенсора в отдельности, а работать уже с физической системой которая объединяет в себе все эти компоненты.

Редактор SensorML написан на основе редактора smle-editor, автором которого является организация “52North”, большая часть исходного кода была переписана и обновлена до новой версии фреймворка Angular 7, поддержка редактора прекращена 2 года назад (2018) от текущей даты [14]. Бекенд часть позволяет управлять пользовательскими данными и данными о системе (документация, ознакомительная информация). Взаимодействие с платформой происходит через внутренний программный интерфейс платформы.

Проектирование и конструирование модуля платформы виртуализации (Microservices)

Данная платформа написана с применением подхода микросервисной архитектуры [10]. Использованы такие решения, как: golang, go-micro, gRPC, Protobuf, mongodb, docker. Платформа виртуализации содржит следующие микросервисы:

Sensor connection handler. Сервис обработки запросов от сенсорных устройств. Находится под нагрузкой, т.к. является основной точкой входа в систему. При необходимости для снижения общей нагрузки на систему можно поднять несколько копий на разных серверных машинах. Система с одним экземпляром данного сервиса способна обработать около 500 запросов в секунду от сенсорных устройств.

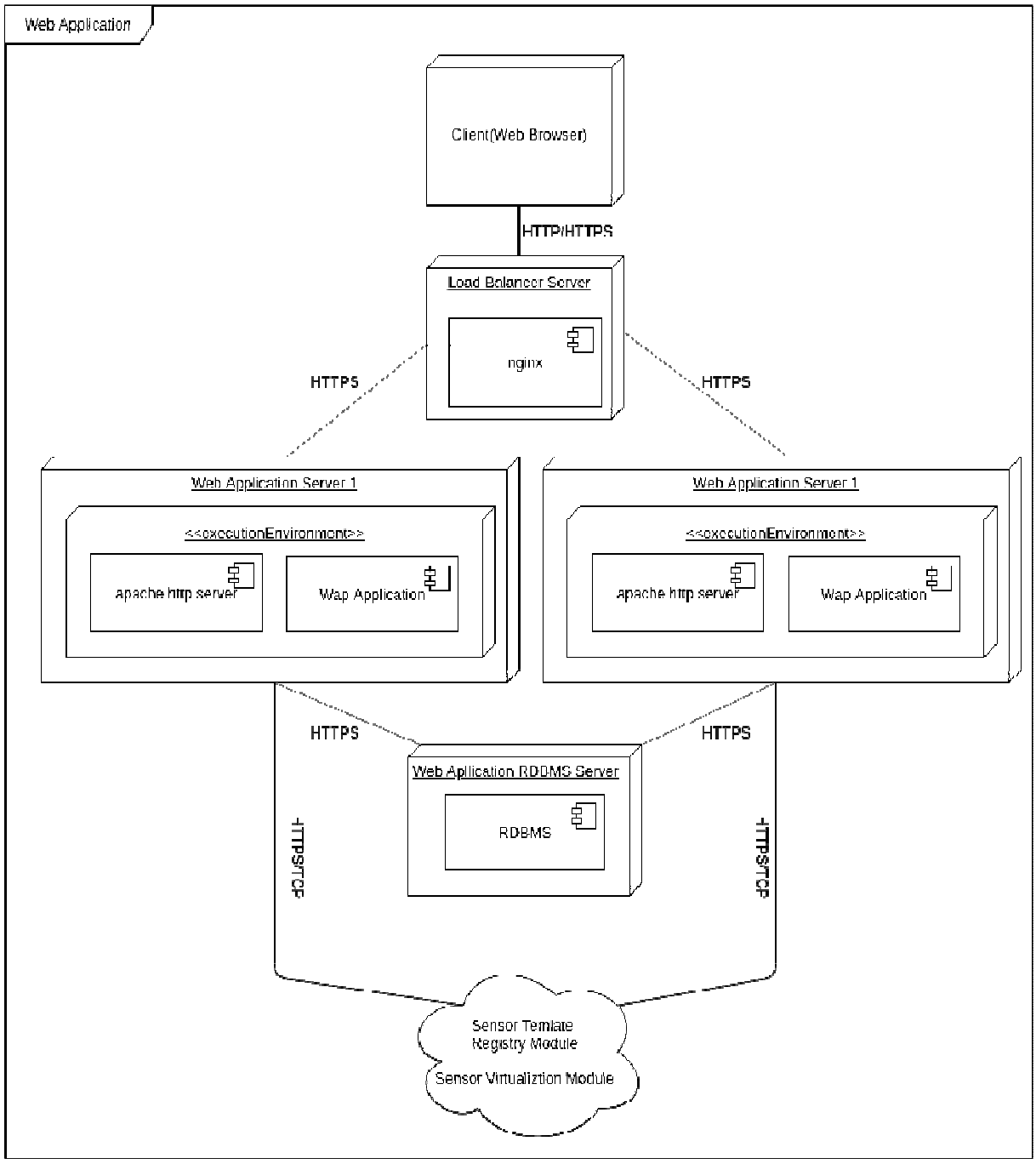

Рисунок 7 – Физические узлы веб приложения (диаграмма развертывания)

Sensor Resolver. Сервис маршрутизации сенсор – контейнер. По своему устройству напоминает работу свитча(switch) – устройства для маршрутизации в сети. Хранит соотношения сенсор – контейнер с учетом расположения узла – группы виртуальных датчиков и перенаправляет запросы от сервисов Sensor connection handler к узлам, где находится виртуальное представление датчика, пославшего запрос.

Sensor data registry. Общее хранилище (реестр) данных, полученных от всех устройств, зарегистрированных в системе. Является легковесным сервисом, хранение данных осуществляется при помощи базы данных mongodb.

Internal API. Программный интерфейс для взаимодействия бекенда веб-сайта с платформой.

External API. Программный интерфейс для взаимодействия с клиентами системы. Клиенты могут посылать запросы на получение данных за определенный период или подписываться на обновление данных.

Ogc-models. Библиотека с описанием моделей данных. Данная библиотека содержит в себе следующие стандарты OGC: GCO, GMD, GML, SAMS, SF, SML, SWE

Virtual sensor node. Группы виртуальных датчиков;

Node Proxy – сервис, который принимает запросы от Sensor Resolver и перенаправляет их на контейнеры, которые содержат в себе виртуальные представления устройств, данные контейнеры и прокси сервис находятся в одной виртуальной сети.

Sensor Virtual Instance – виртуальное представление устройства, создается на основе sensorML шаблона. Может представлять как физический компонент, так и физическую систему или процесс. Т.к. данный сервис имеет всю информацию о своей физической сущности, он “знает”, как интерпретировать информацию, полученную от физического устройства, кеширует ее и перенаправляет на сервис реестра информации полученной от устройств.

Виртуализация устройства из пользовательского интерфейса. Последовательность событий при виртуализации изображена на рис. 8. При нажатии на кнопку “виртуализировать”, запрос направляется на сервер, где берется шаблон выбранного датчика и запрос посылается в платформу, на внутренний программный интерфейс системы. Данный программный интерфейс перенаправляет запрос на сервис виртуализации. сервис виртуализации создает или пересоздает, если уже существует, контейнер с виртуальным представлением устройства на основе полученного шаблона и возвращает идентификационную строку по тому же маршруту пользователю.

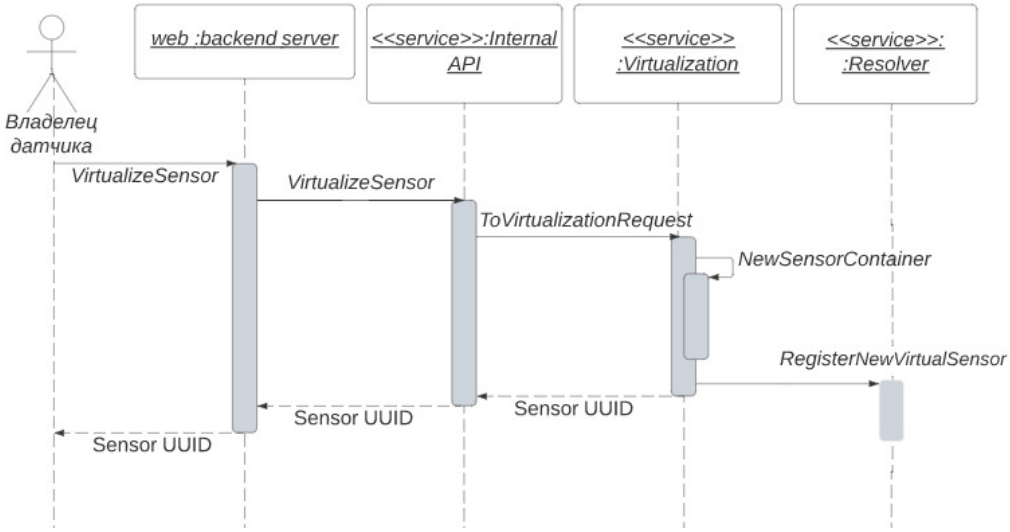

Рисунок 8 – Последовательность действий при запуске виртуализации датчика владельцем

Пользователь, используя эту строку, может идентифицировать свое устройство в системе. Для идентификации ему нужно следовать инструкции, которая будет показана ему вместе с идентификационной строкой. Далее сервис виртуализации добавляет созданный контейнер в виртуальную сеть узла, после чего сообщает резолверу о регистрации нового виртуального представления устройства и все его данные.

Диаграмма развертывания платформы виртуализации

Диаграмма развертывания архитектуры модулей платформы виртуализации датчиков (microservices) (рис. 9) и диаграмма развертывания архитектуры узла виртуальных сенсоров (рис. 10) описывают наиболее типичную конфигурацию

платформы для поставщика услуг “виртуализация датчиков для распределенных измерительных систем”.

Тестирование и оценка производительности прототипа системы виртуализации сенсорных устройств. При разработке системы проводилось тестирование компонентов (модульное тестирование), тестирование интеграции, системное тестирование и приемочные испытания. Приемочные испытания проводились для определения соответствия функциональным требованиям для веб-сервера и платформы виртуализации в соответствии со следующими сценариями:

- Создание сущности – тест проверяет создание сущности в системе.
- Частичное обновление сущности, тест проверяет обновление поля сущности.
- Получение информации, тест проверяет получение информации о физическом датчике и создание виртуальной копии датчика.
- Запрос к физическому устройству, тест проверяет результат запроса для передачи данных от физического датчика его виртуальной копии с последующим их сохранением.

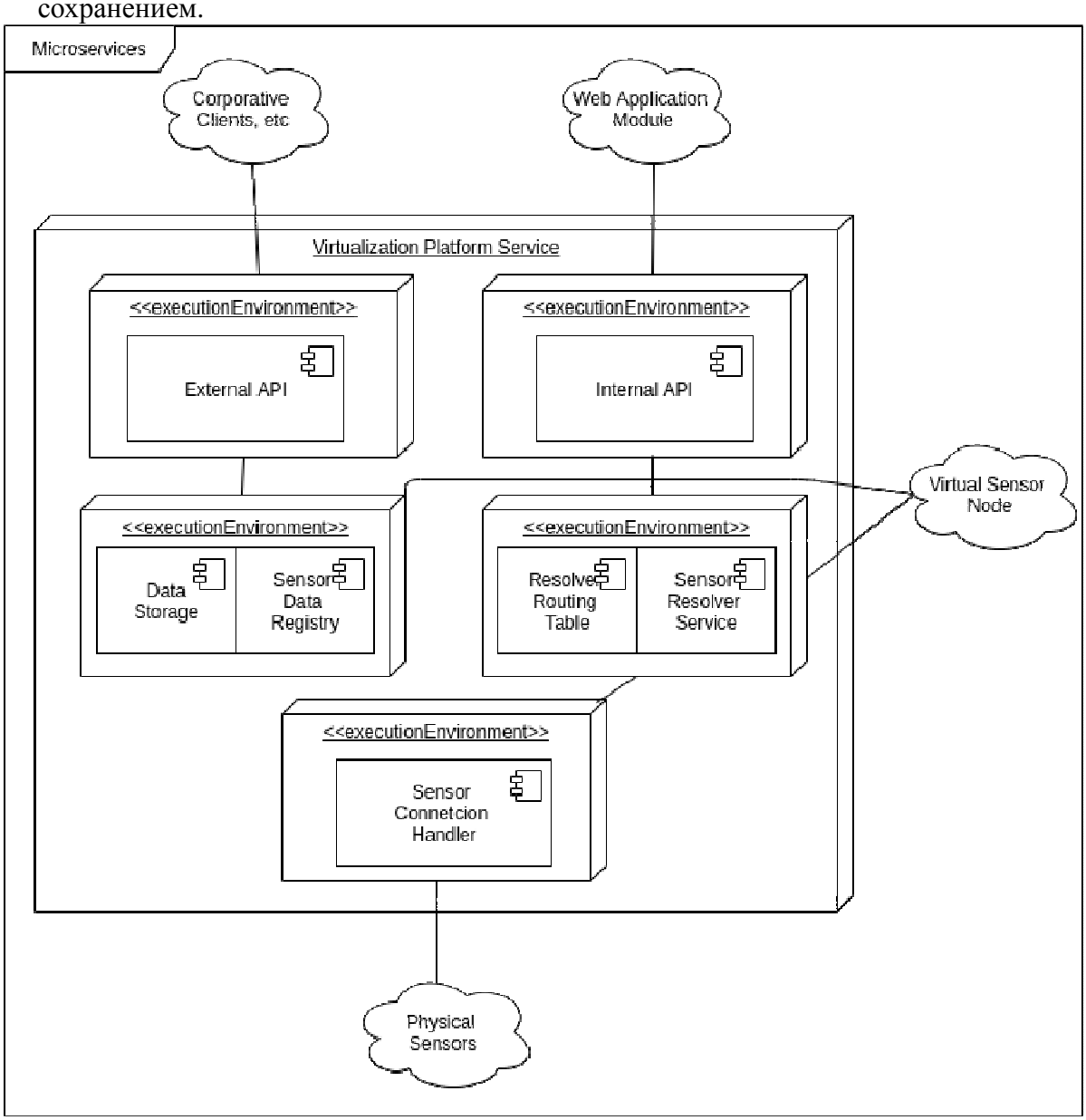

Рисунок 9 – Диаграмма развертывания архитектуры модулей платформы виртуализации датчиков.

При нагрузочном тестировании было выявлено, что один экземпляр сервиса sensor connection handler может выдержать не более 500 запросов в секунду. При возрастании нагрузки на данный сервис требуется установить еще одну копию сервиса и балансировать нагрузку между экземплярами сервиса.

Прототип системы тестировался на параметрах – процессор 2.0GHz, 8GB оперативной памяти и был запущен на десктопной системе с другими потребителями.

При нагрузке до 500 запросов в секунду система потребляет приемлемое количество ресурсов. Стоит сделать замечание, что качество канала связи между датчиком и системой не может быть известно заранее, что может повлечь за собой проблему медленно читающего клиента. Стоит рассмотреть внесение в систему возможности автоматического масштабирования точек входа для датчиков – sensor connection service.

Использование интерфейса не вызвало затруднений. Предъявленные к пользователю требования – знать характеристики своего датчика, являются оправданными. Редактор SensorML является интуитивно понятным для пользователя, знакомого с английским языком и устройством своего датчика.

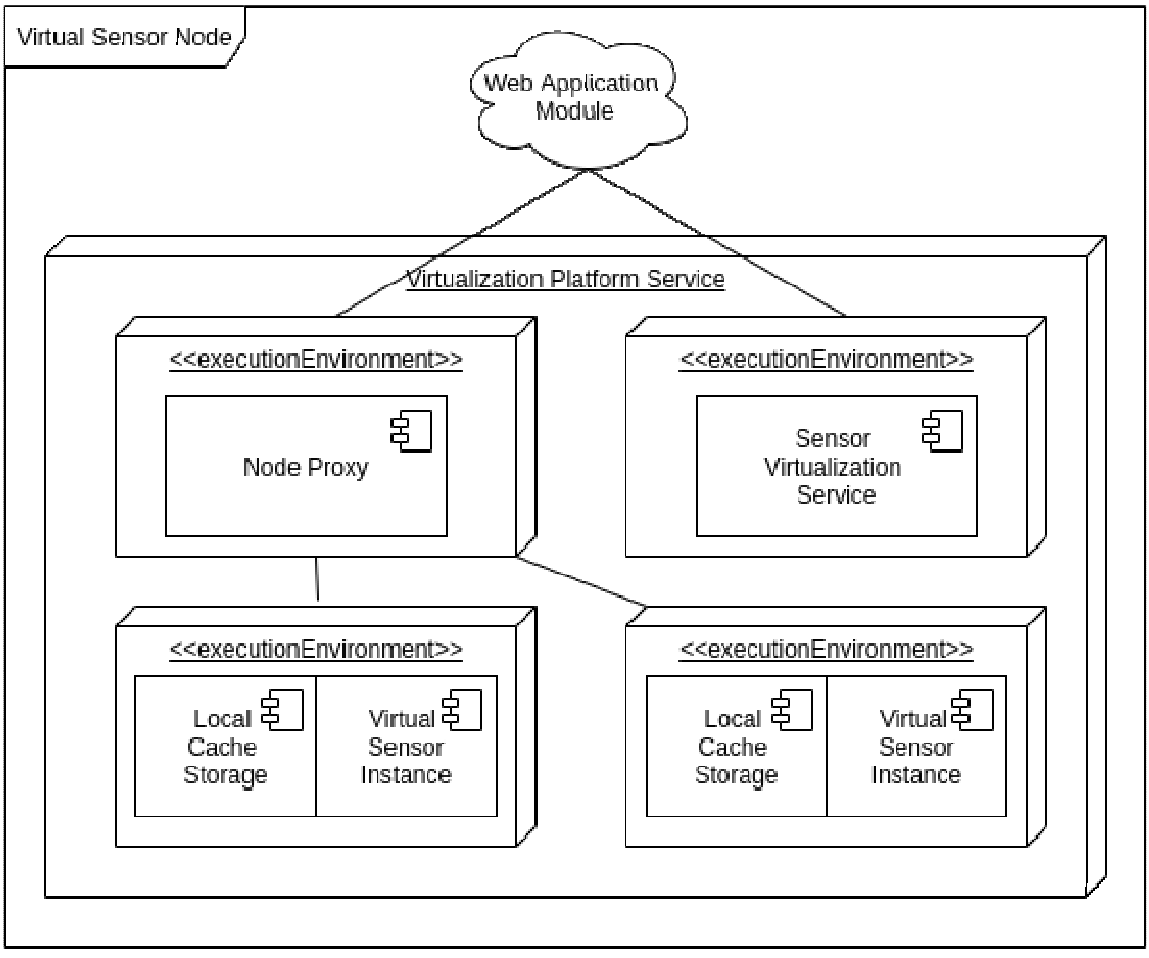

Рисунок 10 – Диаграмма развертывания архитектуры узла виртуальных сенсоров

Заключение

Был разработан и протестирован прототип системы виртуализации датчиков. Смоделирована архитектура системы. Для реализации системы виртуализации датчиков были выбраны языки: Golang, php7.4, typescript в сочетании с популярными

технологиями: go-micro, protocol buffers, grpc, docker. Предложенный подход по виртуализации сенсорных устройств позволил достичь следующего:

- Реализована виртуализация датчиков;
- Система реализует стандарт SensotML и позволяет пользователям легко описать свои устройства, используя веб форму.
- Система обслуживает большое количество запросов (до 500 запросов в секунду на одну точку входа при конфигурации сервера 4Гб оперативной памяти и двухъядерном процессоре с частотой 2.0 GHz).
- Система обеспечивает постоянный доступ к виртуальным представлениям устройств и их данным.
- Система может принимать данные любых типов устройств, описанных при помощи стандарта SensorML.
- Система скрывает физические устройства за слоем абстракции, таким образом, что пользователи не имеют неограниченного доступа к устройствам и не провоцируют разряд батареи устройства высоким количеством запросов, но могут полноценно взаимодействовать с их виртуальными копиями.
- Система позволяет пользователю не обращаться на адрес в сети каждого из устройств, а иметь их представления на одном узле, что облегчает работу с адресацией устройств в сети для пользователя

В дальнейшем планируется наращивать функционал системы для виртуализации датчиков и улучшать возможности масштабирования системы.

Система была использована для исследования эффективности виртуализации датчиков. С ее помощью были проведены эксперименты с использованием технологий виртуализации контейнеров и сетей. Проведен анализ эффективности работы виртуальных устройств с использованием контейнеров, подключаемых к виртуальным сетям и передачи данных между глобальной сетью и слоями виртуальных сетей, получены результаты и описаны дальнейшие перспективы исследования.

Литература

1. Madoka Yuriyama, Takayuki Kushida, Sensor-Cloud Infrastructure – Physical Sensor Management with Virtualized Sensors on Cloud Compputing. 13th International Conference on Network-Based Information Systems 2010.
2. Khan, I.; Errounda, F.Z.; Yangui, S.; Glitho, R.; Crespi, N., – Getting Virtualized Wireless Sensor Networks' IaaS Ready for PaaS / Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference.
3. Atrayee Gupta and Nandini Mukherjee Implementation of Virtual Sensors for Building a Sensor-Cloud Environment Accepted in the 8th International Conference on Communication Systems and Networks (COMSNETS) 2016.
4. M. Yuriyama, T. Kushida, and M. Itakura, A new model of accelerating service innovation with sensor-cloud infrastructure, in Proceedings of the annual SRII Global Conference (SRII '11), pp. 308–314, 2011.
5. Mihui Kim, Mihir Asthana, Siddhartha Bhargava, Kartik Krishnan Iyyer, Rohan Tangadpalliwar, Jerry Gao, Developing an On-Demand Cloud-Based Sensing-as-a-Service

System for Internet of Things // Journal of Computer Networks and Communications, Volume 2016, Article ID 3292783, 17 pages.
6. Гайдамако В.В. Инфраструктура SENSOR-CLOUD – облачные информационно-измерительные системы / Проблемы автоматики и управления. – 2018. – №2 (35). С. 109–118.
7. The Home of Location Technology Innovation and Collaboration. URL: https://www.ogc.org/, (дата обращения 08.06.2020).
8. ISO. Международная Организация по Стандартизации. URL: https://www.iso.ru/, (дата обращения 08.06.2020).
9. https://developer.mozilla.org/ru/docs/Web/XML/ (дата обращения 11.05.2020).
10. Авельцов Д.О. Применение микросервисной архитектуры в разработке программного обеспечения системы мониторинга параметров окружающей среды // Проблемы автоматики и управления. – 2018. – №1 (34). – С. 34–43.
11. Wilde E., Pautasso C. REST: From Research to Practice // Springer Science & Business Media, 2011.
12. URL:https://developer.mozilla.org/ru/docs/Web/HTTP. (дата обращения 11.05.2020)
13. URL:https://www.json.org/ (дата обращения 11.05.2020).
14. URL:52north.github.io/smle/master (дата обращения 11.05.2020).