**Введение**

В настоящее время получили распространение большое количество разнообразных датчиков потребительского назначения для автоматизированных систем, именуемых как «Умный дом». Для конечного потребителя, интересующегося данной сферой и желающего охватить и контролировать с помощью этих устройств максимальное количество аспектов своей жизнедеятельности, возникает непреодолимая преграда в виде шквала разнообразной информации, поступающей от всевозможных датчиков. Разобраться во всем этом порой не под силу даже опытному инженеру. Упорядочить, визуализировать и сделать удобочитаемыми для пользователя призваны средства мониторинга и анализаторы данных.

В загородном доме возникла потребность создания автоматической системы для слежения за состоянием отопления. И для удаленного получения и обработки поступающей с нее информации разрабатывается облачный анализатор данных автоматизированных систем домашнего пользования.

Цель:

* Разработать прототип облачного анализатора данных автоматизированных систем домашнего пользования.

Задачи:

* Провести обзор аналогов облачного анализатора данных для автоматизированных систем домашнего пользования;
* Осуществить выбор средства для разработки анализатора;
* Выполнить разработку базы данных для анализатора;
* Разработать алгоритм анализа данных;
* Разработать клиентскую часть анализатора;
* Подготовить данные для разрабатываемого алгоритма.

**Обзор аналогов**

Чтобы определить функционал разрабатываемой системы, проведем анализ имеющихся на рынке аналогов. Анализатор данных представляет собой средство для сбора и анализа данных, поступающих с различных устройств. К средствам со схожим с ним функционалом можно отнести платформы для управления домашней автоматикой и системы мониторинга сетевых устройств.

Платформы для управления домашней автоматикой – это, иначе говоря, системы «Умный дом», - сочетают в себе не только функции управления всевозможными устройствами, имеющимися в доме, но и предоставляют возможность увидеть в реальном времени состояния тех или иных устройств и показания с различных датчиков, установленных в частных домах. Одними из популярных на данный момент являются системы MajorDoMo, DeviceHive и 1-М Умный дом. Остановимся на них подробнее.



Рисунок 1 MajorDoMo

*MajorDoMo* (см. рис. 1) - представляет собой бесплатную и открытую российскую программную платформу для комплексного управления домашней автоматикой, а так же для информационной поддержки жизнедеятельности. Данная система может быть установлена практически на любой персональный компьютер (на платформе Windows и Linux) и совершенно не требовательна к ресурсам. Даже без привязки к оборудованию она может быть использована для организации персональной инфо-среды. Из интересующих нас его возможностей можно выделить:

* Кросс-платформенность (Windows/Linux);
* Бесплатная для личного или коммерческого использования;
* Поддержка различного оборудования;
* Мульти-язычный интерфейс (Русский/English);
* Веб-доступ с любого устройства;
* Веб-интерфейс с обновлением в реальном времени;
* Интеграция со сторонними веб-сайтами и сервисами;
* CloudSync — облачная синхронизация и простой доступ из любого места;
* Построена на веб-технологиях (PHP/JS/HTML5);
* Программирование с помощью PHP.

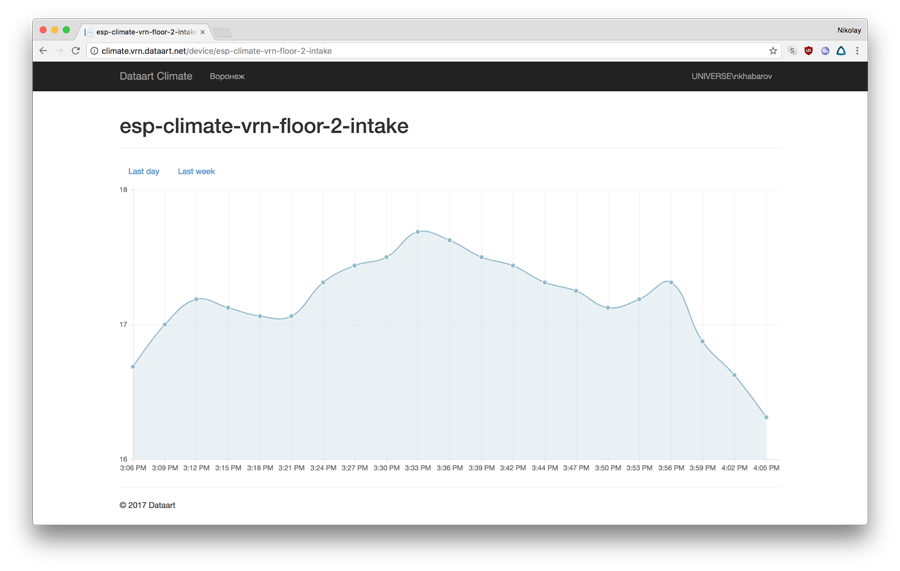
**

Рисунок 2 Пример работы DeviceHive

*DeviceHive* (см. рис. 2) – инструмент, способный сделать частью Интернета вещей любое подключенное к нему устройство. Он обеспечивает коммуникационный уровень, программное обеспечение управления и много платформенные библиотеки для начальной загрузки и разработки умных устройств, домашней автоматизации, дистанционного зондирования, телеметрии, программного обеспечения для дистанционного управления и мониторинга и многое другое. Подключение встраиваемых Linux-систем с использованием Python или C++, протоколов Json. Разработка клиентских приложений с использованием библиотек HTML5/JavaScript, iOS и Android. Из его особенностей выделим:

* Подключение любого устройства через rest API, Websetsets;
* Возможность подключить низкоуровневые устройства с поддержкой Wi-Fi;
* Легкая интеграция с любыми другими облаками и платформами;
* Дает основу для быстрого построения аналитики;
* Открытый исходный код.

*1-M Core* - Программное ядро 1-M Core отечественной разработки превращает Android-устройство, обычный роутер или другое Linux- или Windows- устройство в центральный контроллер Умного Дома. Ядро 1-M Core выполняет прием данных от датчиков, обработку сценариев, управление устройствами и другие функции системы Умного Дома.

Поддерживается радиосвязь с датчиками и устройствами фирм: Livolo, Noolite, RAEX, Telis, TKBHOME, GLANZEN, Chuango, Homesecur, Oregon, Nice, Visonic, AnMotors, CAME, Eazylift, EmilLux, FalconEye, Focus, Gant, Ginzzu, Hormann, Kelvin, Baisheng, TechLiCo, Feron, EmilLux, Expert, Fuhaote и др. Отметим его особенности:

* Оповещения о событиях дома: вторжение, дым, потоп и тп.
* Мониторинг и удаленное управление оборудованием.
* Подключение облачных сервисов: прогноз погоды, геопозиционирование и др.
* Более эффективная, слаженная работа подсистем: отопления, полива, освещения и др.
* Ведение статистики работы устройств на облаке.

Системы мониторинга серверов предназначены для снятия и отображения метрических показателей с различных сетевых устройств. Они предоставляют мощный функционал для системных администраторов и значительно упрощают их работу. Порой на практике их применяют и в целях отображения показаний с датчиков, применяемых в системах домашнего пользования. Остановимся на наиболее используемых системах мониторинга: PRTG Network, Zabbix и Cacti. Опишем приведенные системы мониторинга.

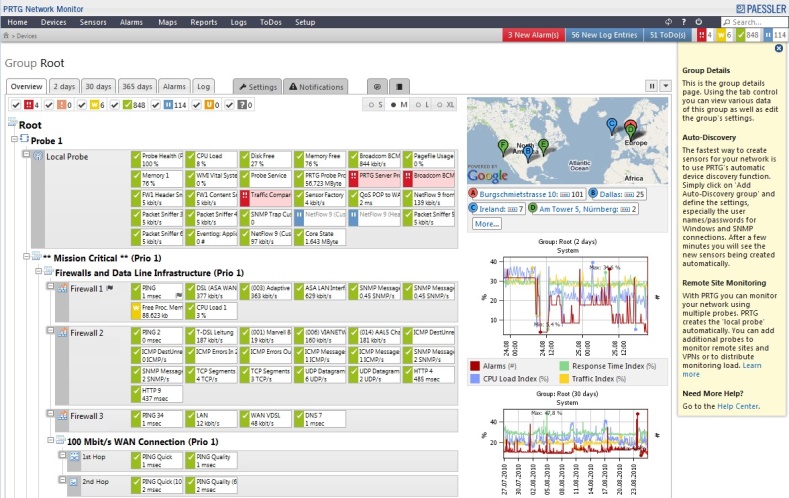


Рисунок 3 Рабочее окно PRTG Network

*PRTG Network* (см. рис. 3) – условно-бесплатная программа (пробный период — 30 дней) предназначенная для мониторинга использования сети, работает в семействе операционных систем Windows. Из возможностей данной программы:

* Сбор информации о потоках данных, проходящих через конкретные устройства, с сохранением её в базе данных программы;
* Просмотр статистики в базе данных в виде графиков и таблиц;
* Просмотр результатов в режиме реального времени или за определенный промежуток времени в прошлом на разных устройствах;
* Сбор данных о нагрузке на подсистемы памяти и процессора.

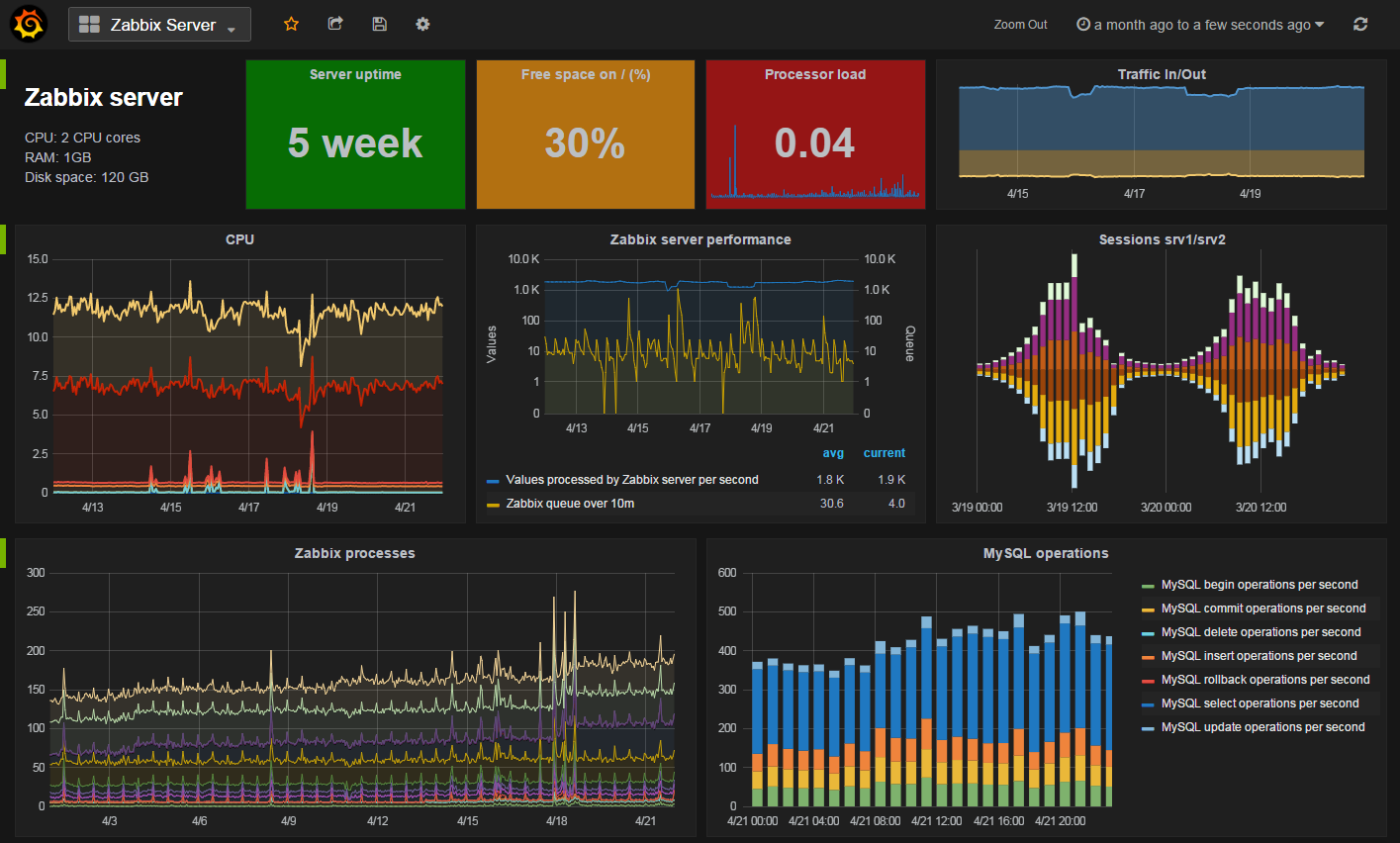


Рисунок 4 Рабочее окно Zabbix

*Zabbix* (см. рис. 4) – свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования, написанная Алексеем Владышевым. Для хранения данных используется MySQL, PostgreSQL, SQLite или Oracle Database, веб-интерфейс написан на PHP. Из возможностей данной программы:

* Распределённый мониторинг — до нескольких тысяч узлов. Конфигурация младших узлов полностью контролируется старшими узлами, находящимися на более высоком уровне иерархии;
* Сценарии на основе мониторинга;
* Централизованный мониторинг журналов;
* Веб-интерфейс для администрирования и настройки;
* Отчётность и тенденции;
* Комплексная реакция на события;
* Расширение за счёт выполнения внешних скриптов;
* Гибкая система шаблонов и групп.

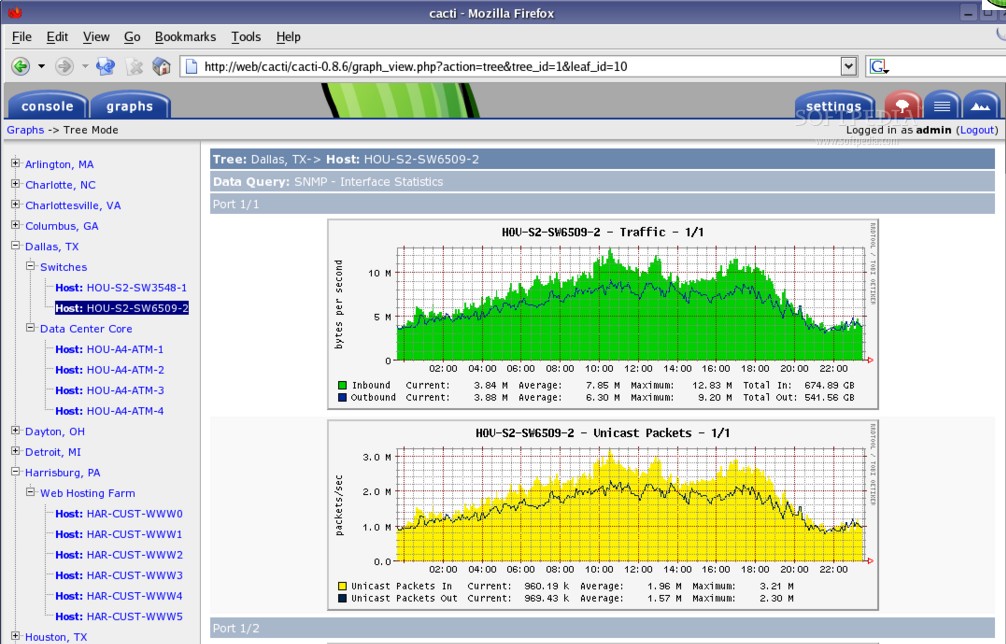


Рисунок 5 Рабочее окно Cacti

*Cacti* (см. рис. 5) – open-source веб-приложение, система позволяет строить графики при помощи RRDtool (набор утилит для работы с кольцевыми базами данных). Cacti собирает статистические данные за определённые временные интервалы и отображает их в графическом виде. Преимущественно используются стандартные шаблоны для отображения статистики по загрузке процессора, выделению оперативной памяти, количеству запущенных процессов, использованию входящего/исходящего трафика. Из возможностей Cacti:

* Неограниченное количество элементов графика может быть определено для каждого графика;
* С помощью математических функций CDEF (позволяет применять различные математические функции к графику для изменения выходных данных) можно управлять данными графа встроенными в RRDTool. Эти функции CDEF можно определить в Cacti и использовать глобально на каждом графике;
* Параметры RRD настраивают для пользователя возможность сбора данных по нестандартным временным интервалам при одновременном хранении различных данных;
* Содержит механизм ввода данных, который позволяет пользователям определять пользовательские сценарии, может быть использовано для сбора данных;
* Шаблоны графиков позволяют группировать общие графики по шаблону.

Для того чтобы провести оценку приведенных ранее приложений сперва выделим те качества, которые нас интересуют, а именно: стоимость приобретения, страну происхождения, место развертывания (ПК, облако, физический сервер), возможность строить графики из поступающей информации, возможность обрабатывать поступающие данные, возможность делать прогнозы и количество подключаемых датчиков. Достоинства и недостатки приведенных выше приложений представлены в таблице 1.

*Таблица 1.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | MajorDoMo | DeviceHive | 1-М Core | PRTG Network | Zabbix | Cacti |
| Стоимость | Бесплатно | Бесплатно | Бесплатно | Бесплатно на 30 дней | Бесплатно | Бесплатно |
| Страна | Россия | США | Россия | Германия | Латвия | США |
| Расположение | ПК | Облако | ПК | ПК | Физический сервер | Физический сервер |
| Наличие графика | Присутствует | Отсутствует | Отсутствует | Присутствует | Присутствует | Присутствует |
| Обработка данных | Отсутствует | Присутствует | Отсутствует | Присутствует | Присутствует | Присутствует |
| Наличие прогноза | Отсутствует | Отсутствует | Отсутствует | Отсутствует | Присутствует | Отсутствует |
| Количество датчиков | Не ограничено | Не ограничено | 50 | 30 | Не ограничено | Не ограничено |

Нетрудно сделать вывод, что большинство из вышеуказанных средств только отображают поступающую информацию с датчиков и отмечают значения, отклоняющиеся от заданных границ. Лишь PRTG Network, Zabbix и Cacti обладают достаточным функционалом, чтобы должным образом анализировать поступающую информацию и даже давать краткосрочные предсказания по значениям поступающих данных (Zabbix). Но их недостатками является то, что компании разработчики расположены за пределами РФ и их продукты разворачиваются не в облаке, а на физических серверах, следовательно, доступ к их данным возможен через сеть Интернет только после договоренности с провайдером.

**Требования к разрабатываемому облачному анализатору данных автоматизированных систем домашнего пользования**

Облачный анализатор данных автоматизированных систем домашнего пользования представляет собой средство для сбора, агрегации и анализа данных, поступающих через сеть Интернет с различных датчиков. Он расположен на сервере и взаимодействует с пользователем через браузер или специальное приложение. В нем реализована возможность долговременного хранения информации, ее обработки, отображения в реальном времени поступающей информации с датчиков. Он способен постоянно анализировать информацию и с учетом вновь поступающей производить переоценку сделанных ранее выводов. Стоит отдельно отметить наличие особого алгоритма, который способен обнаруживать в потоке поступающих данных отклонения, не характерные для основной массы поступающей информации и именуемые далее «аномалиями». Так же этот алгоритм может идентифицировать такие аномалии из числа других аномалий по характерным «сигнатурам» – наличию, величине, количеству и продолжительности экстремумов в поступающей информации, а так же после возникновения первых отклонений от нормы в поступающей информации предсказывать определенные аномалии с вычисленной вероятностью их появления.

Функциональные требования:

1. Получение и запись в БД данных датчиков, поступающих через сеть по протоколу HTTP;
2. Добавление нового датчика в анализатор;
3. Информация о датчиках хранится в отдельной таблице БД;
4. Возможность идентифицировать конкретный датчик по данным, поступающим с него;
5. Возможность создавать кластеры данных, поступающих из определенных датчиков;
6. Информация, определяющая кластеры данных, хранится в отдельной таблице БД;
7. Долговременное хранение в БД данных с датчиков;
8. Хранит информацию об ошибках в отдельной таблице;
9. Отображение всех имеющихся в БД данных;
10. Отображение данных из БД за определенный срок;
11. Отображение всех данных из БД одного датчика;
12. Отображение из БД данных определенного датчика или группы датчиков за определенный срок;
13. Отображение всех аномалий за весь период времени на графике;
14. Отображение местоположений конкретной аномалии на графике;
15. Информация о местоположениях конкретной аномалии на графике хранится в отдельной таблице;
16. Информирование об обнаружении аномалии;
17. Отображение вероятности идентификации новой аномалии при ее обнаружении как той или иной уже сохраненной;
18. Автоматическое определение новой аномалии;
19. Аномалия определяется только для определенных кластеров данных;
20. Отображение подробной информации о конкретной аномалии;
21. Отображение графика в выводе;
22. Постоянное обновление данных в выводе;
23. Отображение линии тренда на графике, рассчитанного для данных за определенный период времени;
24. Отображение регрессии на графике, рассчитанного для данных за определенный период времени;
25. Отображение доверительного интервала на графике, рассчитанного для данных за определенный период времени;
26. Возможность предсказывать на ограниченный срок значения, поступающих с датчиков данных;
27. Доступ к функционалу анализатора через авторизацию;
28. В анализаторе имеется 2 группы пользователей: администраторы и клиенты;
29. Хранение логинов и паролей клиентов и администраторов в отдельных таблицах БД;
30. Разные клиенты не могут обратиться к одинаковым данным БД;
31. Группа «Клиенты»:
    1. Имеет возможность зарегистрироваться;
    2. Может авторизоваться;
    3. Может добавлять и удалять датчики, с которых поступает информация;
    4. Может делать вышеуказанные выборки данных из БД;
    5. Получать анализ данных;
    6. Имеет возможность описывать обнаруживаемые аномалии;
    7. Имеет возможность удалять сигнатуры аномалий;
32. Группа «Администраторы»:
    1. Может авторизоваться;
    2. Имеет возможность создавать новых администраторов;
    3. Может удалить администратора;
    4. Может удалить клиента и привязанные к нему данные;
    5. Может добавить нового клиента;
    6. Имеет возможность просматривать записи из всех таблиц БД;
    7. Имеет возможность удалять записи из всех таблиц БД;
    8. Имеет возможность читать информацию об ошибках;
    9. Имеет возможность добавлять свои сигнатуры аномалий;
    10. Имеет возможность удалять сигнатуры аномалий;
    11. Имеет возможность описывать обнаруживаемые аномалии;
    12. Может вносить изменения в параметры работы алгоритмов анализа;
    13. Может делать вышеуказанные выборки данных из БД;
    14. Получает анализ данных;

Нефункциональные требования:

1. Время отклика анализатора не более 20 с.;
2. Анализатор должен быть отказоустойчивым;
3. Данные в БД хранятся не более 5 лет;
4. Анализатор должен предоставлять удобный и понятный пользовательский интерфейс;
5. Анализатор предоставляет возможность для одновременной работы 20-ти пользователей

**Разработка архитектуры облачного анализатора данных автоматизированных систем домашнего пользования.**

Формирование сущностей и атрибутов, построение схемы модели базы данных.

Согласно требованиям была составлена таблица сущностей и их атрибутов, которые будут применены в БД анализатора (см. табл. 2).

Таблица 2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Название сущности** | **Название атрибута** | **Назначение атрибута** | **Описание атрибута** |
| Data | DataID | Идентификатор записи | Величина типа bigint. Первичный ключ. Номер записи. Не пустое. |
| Time | Время измерения | Поле типа datetime. Содержит в себе время измерения показаний. Не пустое. |
| Value | Измеренное значение | Поле типа float. Содержит в себе величину, измеренную датчиком. Не пустое. |
| SensorID | Идентификатор датчика | Поле типа int. Внешний ключ. Не пустое. |
| Sensors | SensorID | Идентификатор датчика | Величина типа int. Первичный ключ. Номер датчика. Не пустое. |
| Name | Имя датчика | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Description | Информация о датчике | Величина типа varchar. Содержит не более 1000 символов. Может быть пустым. |
| ClusterID | Принадлежность кластеру | Величина типа int. Внешний ключ. Может быть пустым. |
| MaxValue | Максимальная возможная величина, исходящая от датчика | Величина типа float. Может быть пустым. |
| MinValue | Минимальная возможная величина, исходящая от датчика | Величина типа float. Может быть пустым. |
| ClientID | Идентификатор клиента | Поле типа int. Внешний ключ. Не пустое. |
| ValidInterv | Определяет размер допустимого интервала | Поле типа float.  Может быть пустым. |
| IntervalPeace | Определяет продолжительность состояния покоя | Поле типа int  Может быть пустым. |
| Clusters | ClusterID | Идентификатор кластера | Величина типа int. Первичный ключ. Номер кластера. Не пустое. |
| Name | Имя кластера | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Description | Информация о кластере | Величина типа varchar. Содержит не более 1000 символов. Может быть пустым. |
| Client | ClientID | Идентификатор клиента | Величина типа int. Первичный ключ. Номер клиента. Не пустое. |
| Login | Логин клиента | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Password | Пароль клиента | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Administrators | AdministratorID | Идентификатор администратора | Величина типа int. Первичный ключ. Номер администратора. Не пустое. |
| Login | Логин администратора | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Password | Пароль администратора | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Logs | LogID | Идентификатор записи в журнале ошибки | Величина типа int. Первичный ключ. Номер записи в журнале. Не пустое. |
| TimeRegister | Время регистрации ошибки | Поле типа datetime. Содержит в себе время регистрации ошибки. Не пустое. |
| Description | Описание ошибки | Величина типа varchar. Содержит не более 1000 символов. Не пустое. |
| Anomalies | AnomalyID | Идентификатор аномалии | Величина типа int. Первичный ключ. Номер аномалии. Не пустое. |
| TimeRegister | Время регистрации аномалии | Поле типа datetime. Содержит в себе время регистрации аномалии. Не пустое. |
| Name | Название аномалии | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Description | Информация об аномалии | Величина типа varchar. Содержит не более 1000 символов. Может быть пустым. |
| ClientID | Идентификатор клиента | Поле типа int. Внешний ключ. Не пустое. |
| Signatures | SignatureID | Идентификатор позиции в сигнатуре | Величина типа bigint. Первичный ключ. Номер позиции в сигнатуре. Не пустое. |
| Value | Величина позиции сигнатуры | Величина типа float. Не пустое. |
| AnomalyID | Идентификатор аномалии | Поле типа int. Внешний ключ. Не пустое. |
| Labels | LabelID | Идентификатор метки | Величина типа int. Первичный ключ. Номер метки. Не пустое. |
| Start | Время начала метки | Поле типа datetime. Содержит в себе время начала метки. Не пустое. |
| End | Время конца метки | Поле типа datetime. Содержит в себе время окончания метки. Не пустое. |
| AnomalyID | Идентификатор аномалии | Поле типа int. Внешний ключ. Не пустое. |

После построения списка сущностей была построена схема модели базы данных. Сущности были представлены в виде таблиц с полями, а отношения составили связи между ними. Схема модели была построена с помощью Microsoft Visio. Ниже приведена схема модели (см. рис. 1).

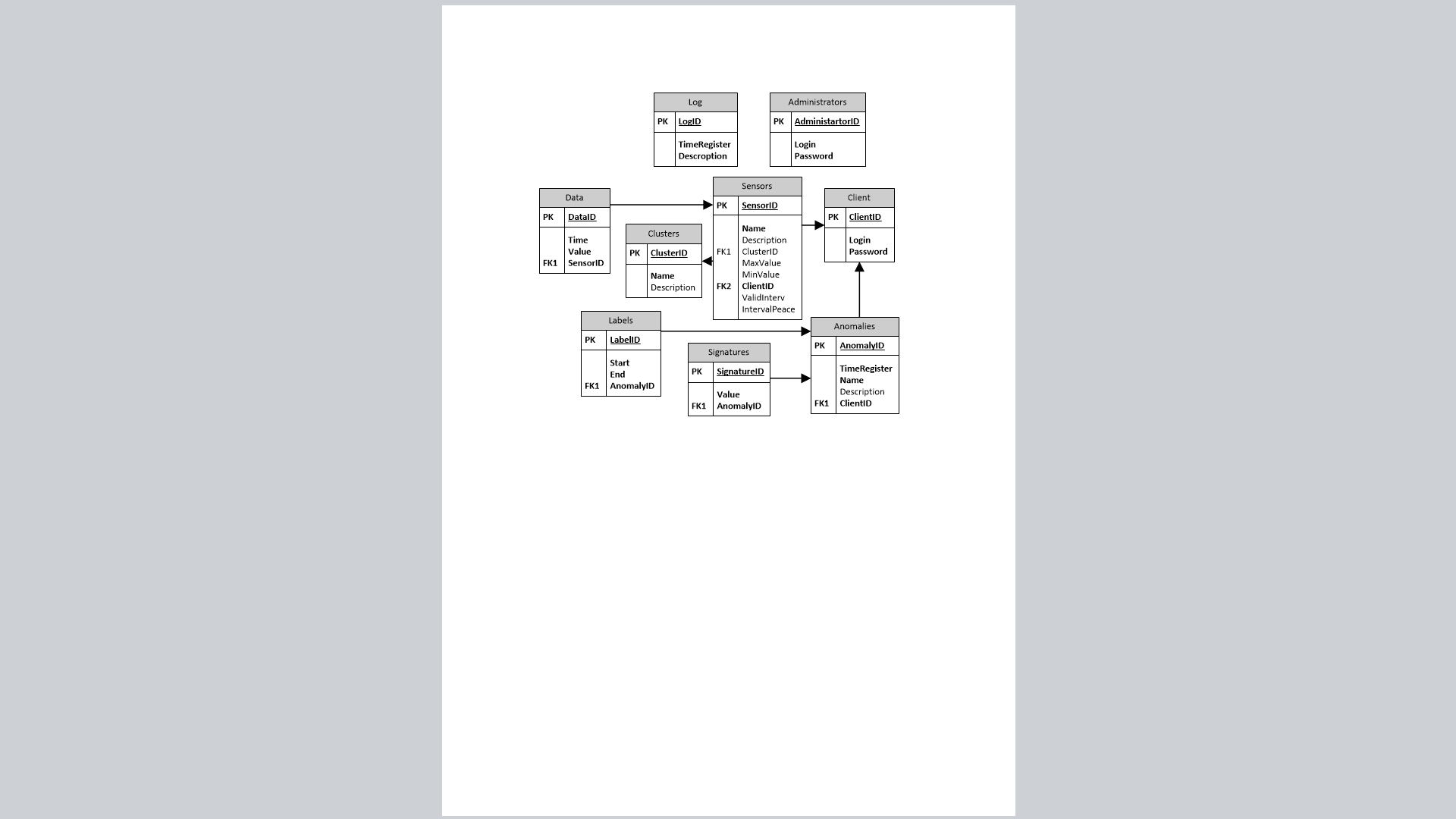


Рисунок 1. – Схема модели базы данных